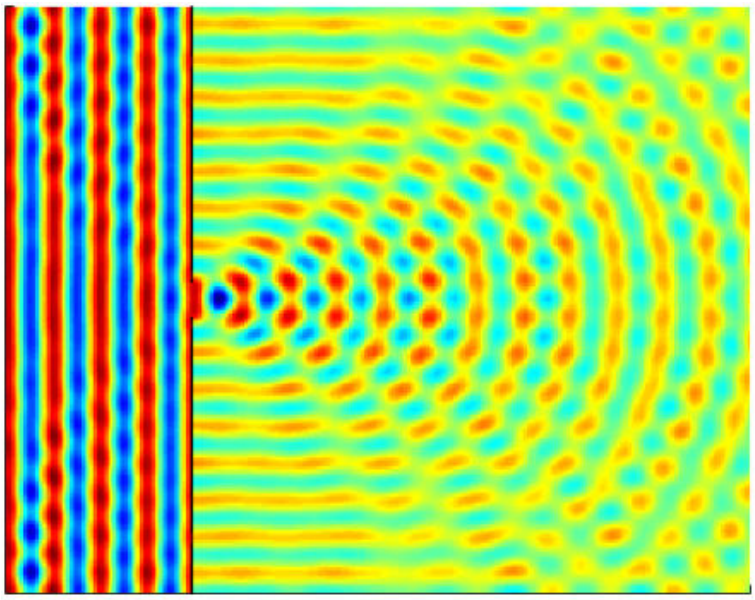
Master of Science Thesis



Mimetic Discretization of the High Frequency Helmholtz Equation using Hermite Interpolating Polynomials

Peter Klaas Christiaan Lok

February 22, 2020

Aerodynamics

Faculty of Aerospace Engineering

TU Delft

Delft University of Technology

Mimetic Discretization of the High Frequency Helmholtz Equation using Hermite Interpolating Polynomials

Master of Science Thesis

For obtaining the degree of Master of Science in Aerospace Engineering
at Delft University of Technology

Peter Klaas Christiaan Lok

February 22, 2020

Faculty of Aerospace Engineering · Delft University of Technology

Delft University of Technology

DELFT UNIVERSITY OF TECHNOLOGY
DEPARTMENT OF AERODYNAMICS

The undersigned hereby certify that they have read and recommend to the Faculty of Aerospace Engineering for acceptance the thesis entitled **"Mimetic Discretization of the High Frequency Helmholtz Equation using Hermite Interpolating Polynomials"** by **Peter Klaas Christiaan Lok** in fulfillment of the requirements for the degree of **Master of Science**.

Dated: February 22, 2020

Supervisors:

Dr.ir. M.I. Gerritsma

Prof.dr. D.G. Simons

Dr.ir. M. Möller

Y. Zhang, Msc.

Preface

After completing the courses and tests preceding this point in a study program, a thesis is the project it all comes down to. However, writing a thesis has proven to be much more than conducting research and drawing conclusions. Personally, this project has been a roller-coaster. A roller-coaster without fast carts, loops or spectacular turns, but a tedious slow ride with many breakdowns along the way. However, the finish line has been reached and to that extent I would like to thank a number of people. In the first place my supervisor Marc Gerritsma for his patience and advice during this process. At critical points along the way he convinced me to keep going and not give up, thank you. Secondly I would like to thank my friends who kept their interest; in particular thanks to Barend-Jan van Bruchem who helped me with the organization and planning. Thirdly, I would like to thank my parents for keeping faith and providing me with the right perspective. Lastly, and most importantly, I would like to thank my wife, Marja. Your trust, patience, encouragements and hard-working have led me to this point, thank you.

Peter Lok
Gouda, Februari 22, 2020

Summary

Numerical modelling of high frequency waves is a complex and challenging area. Although the underlying equation seems simple, $-\Delta\phi - k^2\phi = f$, the numerical challenges are not. This time harmonic wave equation is known as the Helmholtz equation. The main challenge to be studied is the pollution error, which is the difference between the actual and numerical wavenumber. Due to this error, the solution of most numerical methods deteriorates rapidly when increasing the wavenumber. So far, either problem specific solutions have been found or solutions which require knowledge of the solution itself.

In this thesis a numerical method is developed to approach this challenge. A mimetic discretization of the Helmholtz equation using C^1-continuous Hermite interpolating polynomials is developed to model the Helmholtz equation in 2 dimensions. Mimetic theory is briefly introduced after which the Hermite polynomials are defined and analysed for their interpolating properties. A least-squares variational problem is defined and discretized using 49 basis function at its lowest order. The model is verified using a single sinusoidal wave after which plane wave problems are solved and analysed for their wavenumber-dependence. A diffraction and interference problem is set-up and compared to analytical solutions.

The proposed method shows no significant advantages over the use of C^0-continuous Lagrange polynomials in terms of effectiveness. Both methods show deterioration at the same wave numbers and equal polynomial order. The Hermite polynomials require less unknowns to reach the same accuracy. In short, a p-accurate solution is found using Hermite polynomials at the cost of a $p-1$ model using Lagrange polynomials. This benefit in efficiency does not outweigh the additional effort and boundary information necessary to construct the problem. However, the variety of coefficients offer an opportunity for further research to find relations to the numerical wavenumber in the search of a wavenumber conserving discretization.

Table of Contents

List of Figures

Chapter 1

Introduction

The Helmholtz equation (1.1) is known as the static or time-harmonic wave equation. The equation is fundamental to all wave-related problems such as propagation, reflection, diffraction and interference. These problems are found in many applications such as radar, sonar, seismology, electromagnetics, noise scattering or inverse scattering problems. The latter can be described as a problem where the scattering field and incidence wave(s) are given and it is required to determine the shape of the object. A second application of the Helmholtz equation can be found in the field of quantum mechanics, where it is derived from the Schrodinger equation [29]. In this case the solution represents the probability distribution of the spatial location for a particle at a fixed energy.

$$\nabla\phi + k^2\phi = -f \tag{1.1}$$

At first sight, the Helmholtz equation does not seem like a complex equation. The equation is linear and analytical solutions can easily be found for various problems. When moving to more complicated domains and boundary conditions numerical computation of the solution is required. Due to the highly oscillatory nature of the solutions, especially at high wave numbers, stability and efficiency become a challenge. In order to model, for example, electromagnetic scatter from an airplane the scale differences range from the size of an aircraft to a fraction of the wavelength of the electromagnetic waves. This problem has been stated as one of the most challenging problems in scientific computation by Zienkiewicz [35]. The main numerical challenge in high frequency Helmholtz problems is the so-called pollution or aliasing error. Together with a truncation error and an interpolation error, these three components make up the total error made. This pollution error is encountered as a difference in the numerical (k_n) and physical wavenumber (k). Babuska and Sauter state that this error can be reduced, but is unavoidable in finite element methods (FEM) for problems involving more then 1 dimension [3]. Numerical formulations of the Helmholtz problem usually lead to an indefinite set of matrices. This indefiniteness requires that the mesh size stays below a certain critical value to control the error. In case of the Helmholtz problem, this critical mesh size, or number of points per wavelength, increases with the frequency and is known as pollution [26].

As will be discussed in section 1.2 many methods have been applied to model the equation as effectively and efficiently as possible. Among these methods are finite difference, finite elements, spectral (element) methods, geometrical optics, Trefftz-based methods and more. However, so far no method has matured enough to be widely applied. Therefore a new angle is proposed by using the relatively novel method of mimetic discretization and the determination of explicit derivatives by using Hermite interpolating polynomials. In mimetic theory the fundamental operators of divergence, gradient and curl are discretized exactly, hereby 'mimicking' their behaviour [6]. The generalized Stokes' theorem relates the integrals of differential forms to boundary values, thereby allowing an exact discretization of the operator in an integral formulation [25]. Additional to an exact representation of the differential operators, mimetic discretization techniques have shown to posses high stability and convergence. In addition to using mimetic theory, it is proposed to use Hermite interpolating polynomials as basis functions [10]. This method is closely related to the field of compact finite difference (CFD). In this field the explicit determination of derivative values has proven to increase the order of the discretization, hereby approaching the exponential order of spectral methods. In finite difference higher order schemes are achieved by increasing the number of neighbouring point to determine derivatives. In CFD relations are established that mimic the global dependence of derivative values, by only using the neighbouring cells, hence compact [21]. The goal of this thesis is to examine the effect of using the proposed methods in a finite element formulation of the Helmholtz equation for high frequencies by developing a solver and comparing the results for different example problems to literature.

1.1 Research questions

In the proposed research, the high frequency Helmholtz equation will be modelled using mimetic discretization techniques with Hermite interpolating polynomials. From literature it is expected that the increased stability and convergence rates of the underlying methods will increase the ability to efficiently and effectively solve the problem.
The main aim of this research is to investigate the added value of the mimetic theory in combination with explicit calculation of derivatives in solving the Helmholtz equation for high frequencies. This aim will be achieved by answering the following research questions. Answering these questions will results in overview of the achieved results as well as provide an answer to which extent the goal is achieved. The research questions are stated as follows.

1. Does a numerical mimetic discretization using Hermite interpolating polynomials efficiently and effectively model the Helmholtz equation for high frequencies?
   (a) What are the error sources of the deviations encountered?
   (b) Is the interpolation error for Hermite basis functions reduced with respect to Lagrange basis functions?
   (c) Does the solver reduce pollution errors occurring at high frequencies with respect to general Galerkin FEM?
   (d) Is the computation time necessary to achieve results with low pollution errors smaller compared to general Galerkin FEM?

(e) What are the limitations of the least-squares formulation of the Helmholtz problem and how do they reflect in the results?

The performance of the underlying techniques to be used in this project have shown great promises when applied to other equations and formulations. The stability and convergence rates found in other researches ensure that this research is promising and feasible. Also, the Helmoltz equation is a very fundamental and widely encountered equation as explained in the introduction. Additional research in this field is therefore relevant and necessary.

1.2 Literature Review

1.2.1 Finite Element Methods

General (Galerkin) finite element methods use a weak integral formulation of the Helmholtz problem to construct a system of equations solving the problem. The accuracy and convergence is controlled by either reducing the element size, h, or increasing the polynomial order of the basis functions, p. The relation between these parameters and the wave number, k, has been investigated extensively by Melenk [22]. The research shows that the pollution error reduces significantly with increasing order. Also, explicit bounds, depending on h, p and k are determined to achieve quasi-optimality of the FEM schemes.

The finite element method is inextricably linked to pollution errors. Therefore many improvements have been proposed over the years besides increasing the order. A wide variety of methods have been mentioned in recent overview papers [26] [34]. The most important and relevant methods will be discussed. At first methods to optimize the finite element process are widely used. These include using error estimates, more efficient integral evaluations and improving the solution steps for the system of equations. These methods intend to improve speed and thereby allowing for a finer grid at high frequencies, but do not tackle the source of the problems.

Secondly a variety of methods have been developed to stabilize the solution process, this is achieved by adapting the weak form of the problem. Examples are the quasi-stabilized finite element method (QSFEM), Galerkin least-squares finite element method (GLSFEM) and Galerkin gradient least-squares finite element method (GGLSFEM). As stated before, the indefiniteness of the Helmholtz operator induces the main challenges at hand. Moiola and Spence, however, state the following:

> "whereas the standard variational formulations of the Helmholtz equation are sign-indefinite, this sign-indefiniteness is not an inherent feature of the Helmholtz equation, only of its standard formulations" [23].

Therefore, adapting the weak formulation of the Helmholtz problem can be used to solve the challenges. Recent work by Demkowicz shows that such an alternative formulation results in

a strong reduction of the pollution error [9]. Already in 1979 it was noted that the use of least-squares formulation has a positive effect on the indefiniteness of the resulting operator [13].

The following two classes of methods both require information about the solution beforehand. In the first place multi-scale methods are used, which separate the solution into large and small scales. The smaller scales are approximated using wave function based on a priori knowledge of the solution. These methods show improved accuracy for high frequency problems. However, increased complexity and decreased generality due to the required information of the solution make that the methods are not widely used [12] [20].

The last group of methods are the generalised methods which use knowledge about the solution to reduce the pollution error. These method adapt the basis functions and elements using information about the homogeneous solution. The most well-known method is the partition-of-unity finite element method (PUFEM) [2] [1]. Also Trefftz-based methods, which will be discussed in section 1.2.3 can be considered to be part of this group [18].

1.2.2 Boundary Element Methods

The boundary element method (BEM) uses Green's third identity to relate the computational domain to the boundary using the Green's function and thereby reducing the number of dimensions of the model by 1. In case of the Helmholtz equation, this can be very beneficial in treating unbounded domains. The resulting system of equations in BEM formulations is much smaller compared to finite element methods, however the density of the matrices is much higher and therefore more difficult to solve. In most problems the time gained due to the reduction in dimensions, and therefore in variables, is compensated for by the time needed to construct the problem [26]. Similar to FEM, BEM is unable to accurately solve frequencies which are close to, or correspond to, eigenvalues of the Helmholtz operator. Additionally BEM is also unable to solve the equation at frequencies corresponding to the eigenfunctions of the homogeneous integral equation resulting in BEM formulations [4]. Recent attempts have shown that the approximation error may be bound independently of the wave number using BEM for certain specific geometries. Additionally a condition is posed to bound the approximation error for general domains, which confirms the pollution effect [16].

1.2.3 Other methods

Trefftz based methods

Trefftz based methods construct the solution from basis functions which are solutions to the homogeneous equation itself, so a priori knowledge of the solution is used to construct the model. In this way the equation is satisfied with no residual error, however, the boundary conditions are violated and therefore induce an error in the solution. The method minimizes the residual of these boundary errors. As compared to general finite element methods, Trefftz

based methods are faster, more accurate and able to model high frequencies. The downsides, however, are that the method is more difficult to construct, contains fully populated matrices and most important, is limited to moderately complex, convex, geometries [34] [26].

Spectral methods

Spectral methods use a trigonometric basis function which are non-zero over the entire domain, this results in a robust method with exponential convergence order. The method is however very limited to simple domains. Due to te oscillating nature of the basis function, Gibbs phenomenon plays a role at the boundaries. This results in diverging oscillations and therefore reduced accuracy near the boundaries. Smoothness of the boundary and solution are requirements for the spectral method, which makes its applications limited [15]. However, the properties of convergence and accuracy have inspired the development of other methods such as spectral elements and compact finite difference [21].

Geometrical Optics methods

Method based on geometrical optics (GO) are a more recent development. The basic principle is to decompose the solution in a function for the amplitude, $A(x, k)$, and a function for the phase, $\phi(x)$ [11] [27] and solve the resulting equations separately. Both functions show less oscillations for high frequencies, thereby allowing for a much coarser grid. The method is therefore very suitable for high frequency problems, however also limitations apply. The method is unsuitable when the wavenumber is close to the variations in propagation speed. Diffracted waves are also not accurately modelled by the method as well as caustics[1] [30]. GO methods are mainly accurate in high frequency problems and less accurate for lower frequencies [28].

1.2.4 Mimetic Discretizations

The use of mimetic discretizations is relatively new. Therefore no work is present using the mimetic approach to discretize the Helmholtz equation. Various publications, such as [14], can be found claiming to do so, however in this case the 'good' Helmholtz ($-k^2$ in equation 1.1 instead of $+k^2$) equation has been discretized, which makes the problem sign-definite. Gerritsma and Bochev discretized the 'good' Helmholtz equation using a spectral mimetic least-squares method [5]. The method showed good convergence, order $p + 1$, and exact approximation of the fundamental relations. Similar convergence results as well as an exact approximation of fundamental relations where obtained by Palha on the Poisson equation [25].

[1] This occurs when light rays are refracted or reflected by curved objects and accumulate in an area.

1.2.5 Compact Finite Difference

Compact finite difference is closely related to the proposed methods since it adds additional information of the derivative values to the discretization scheme. Therefore observations and conclusions obtained by applying this method to the Helmholtz problem will provide additional information to substantiate expectations. The method has been introduced by Lele in 1992 [21] and since then used in a variety of applications. In 2013 Turkel [33] developed a compact finite difference scheme of 6th order for the Helmholtz equation which shows low pollution due to the high order of the scheme. Earlier Sutmann [31] developed a similar scheme using Dirichlet boundary conditions. Where Turkel uses a variable wavenumber, Sutmann uses a constant wavenumber. Sutmann shows that a fully compact finite difference scheme, where only neighbouring points are used, is of the same order, but of higher accuracy than a general 6th order finite difference scheme, which uses $2 \times 3 = 6$ neighbouring points. A semi-compact scheme, using two neighbouring points, showed even better accuracy. Nabavi et al [24] implemented a similar 6th order CFD scheme in combination with Neumann boundary conditions and compared this to a fourth order finite element scheme. His results show that the higher order CFD scheme was up to 100 faster. Additionally the work by Cordova [7] needs to be mentioned which uses a mimetic compact finite difference scheme to discretize the time dependent wave equation. Results show that this scheme is up to 4 times faster than a normal finite difference scheme of the same order.

1.3 Thesis Outline

This thesis is structured as follows. After the introduction, the research questions are formulated. Next, a literature review is presented of the numerical methods applied to solve the Helmholtz equation as well as as overview of the theory to be applied in this thesis project. Chapter 2 describes the theoretical basis of the proposed method by deriving the Helmholtz equation from the wave equation, deriving the pollution error and describing the fundamental properties of the equation. Next, the mimetic theory is briefly described after which the Hermite interpolating polynomials are introduced and analysed with respect to the interpolation error followed by a brief introduction of the polynomials to mimetic theory.

Chapter 3 presents the derivation of the mimetic discretization of the Helmholtz equation using the hermite polynomials. A natural two-equation model is derived from the theory of acoustical disturbances. Next a least-squares problem is formulated and discretized. The implementation of boundary conditions are discussed and relevant errors are defined. Finally a finite element method is briefly discussed for comparison.

Chapter 4 reports on the results of the model by presenting a verification problem in the form of a single sinusoidal wave. Next, plane wave experiments are discussed which report on the pollution error followed by diffraction and interference model showcasing the applicability of the model. Finally conclusions are summarized and recommendation are discussed.

Chapter 2

Theoretical Framework

In this chapter the theoretical basis for the newly developed model will be discussed. First, the basic derivation of the Helmholtz equation will be stated as well as the derivation of the pollution error resulting in wavenumber dependent phenomena. Secondly the mimetic theory will be stated consisting of differential geometry, algebraic topology and the application of nodal and edge basis functions. Thirdly, the newly introduced Hermite interpolating polynomials will be discussed and analysed for their convergence characteristics.

2.1 Helmholtz equation

In order to understand the basic properties of the Helmholtz equation the derivation as well as the pollution error will be discussed, concluded by an overview of the fundamental properties of the equation.

2.1.1 Derivation

The Helmholtz is mostly known as the time independent wave equation. Therefore the derivation starts by taking the wave equation stated as follows

$$\frac{\partial^2 \Phi}{\partial t^2} - c^2 \Delta \Phi = F(\bar{x}, t) \tag{2.1}$$

where $\Phi = \mathfrak{Re}\{\Phi\} + i\mathfrak{Im}\{\Phi\}$ is the wave amplitude (real part) and phase (imaginary part), c is the wave speed and F the external forcing. Assuming time harmonic behaviour for both Φ

and F, substituting in equation (2.1) gives

$$\Phi(\bar{x},t) = \phi(\bar{x})T(t) = \phi(\bar{x})e^{i\omega t} \tag{2.2a}$$
$$F(\bar{x},t) = f(\bar{x})T(t) = f(\bar{x})e^{i\omega t} \tag{2.2b}$$
$$\phi(\bar{x})\frac{\partial^2 e^{i\omega t}}{\partial t^2} - c^2\Delta\phi(\bar{x})e^{i\omega t} = f(\bar{x})e^{i\omega t} \tag{2.2c}$$

where ω is the wavenumber. Evaluating the differential and eliminating the time dependency

$$-\omega^2\phi(\bar{x}) - c^2\Delta\phi(\bar{x}) = f(\bar{x}) \tag{2.3}$$

rearranging the terms, dividing by c^2 and defining $k = \frac{\omega}{c}$ provides the Helmholtz equation.

$$-\Delta\phi(\bar{x}) - (\frac{\omega}{c})^2\phi(\bar{x}) = \frac{1}{c^2}f(\bar{x}) \tag{2.4}$$

$$-\Delta\phi - k^2\phi = \bar{f} \tag{2.5}$$

where $\bar{f} = \frac{f}{c^2}$.

2.1.2 Pollution Error

The pollution error is derived by applying a 2nd order discretization scheme to the 1 dimensional homogeneous Helmholtz equation as an example as follows

$$\Delta\phi + \left(\frac{\omega}{c}\right)^2\phi = 0 \tag{2.6}$$
$$\frac{\phi_{i+1} - 2\phi_i + \phi_{i-1}}{h^2} + \left(\frac{\omega}{c}\right)^2\phi_i = 0 \tag{2.7}$$

where ϕ_i is the discrete value of ϕ at location x_i and h is the distance between the points. The exact solution of the homogeneous equation is $\phi(x) = A\exp^{i\frac{\omega}{c}\bar{n}x}$, where $\bar{n}$ is the direction of the wave (-1 or 1 in the 1 dimensional case, taking $\bar{n} = 1$). Inserting the exact solution into the discrete formulation using a different wave speed, $\tilde{c}$, and since $x_{i+1} = x_i + h$ gives

$$\frac{A\exp^{i\frac{\omega}{\tilde{c}}x_{i+1}} - 2A\exp^{i\frac{\omega}{\tilde{c}}x_i} + A\exp^{i\frac{\omega}{\tilde{c}}x_{i-1}}}{h^2} + \left(\frac{\omega}{c}\right)^2 A\exp^{i\frac{\omega}{\tilde{c}}x_i} = 0 \tag{2.8}$$
$$\frac{\exp^{i\frac{\omega}{\tilde{c}}h} - 2 + \exp^{-i\frac{\omega}{\tilde{c}}h}}{h^2} + \left(\frac{\omega}{c}\right)^2 = 0 \tag{2.9}$$

Rewriting the exponential terms using Euler's formula as follows $\exp(ix)+\exp(-ix) = 2\cos(x)$

$$\frac{2(\cos(\frac{\omega}{\tilde{c}}h) - 1)}{h^2} + \left(\frac{\omega}{c}\right)^2 = 0 \tag{2.10}$$

and applying a Taylor expansion, $\cos(x) = 1 - x^2/2 + O(x^4)$, to the cosine, resulting in the approximation of the numerical wave speed as follows, where $n = \frac{1}{c}$

$$\frac{2((1 - (\omega\tilde{n}h)^2/2 + O(\omega\tilde{n}h)^4) - 1)}{h^2} + (\omega n)^2 = 0 \tag{2.11a}$$
$$-\tilde{n}^2 + O(\omega^2 h^2 \tilde{n}^4) + (\omega n)^2 = 0 \tag{2.11b}$$
$$-\tilde{n}^2 + O(\omega^2 h^2 \tilde{n}^4) + n^2 = 0 \tag{2.11c}$$
$$\tilde{n}^2 = n^2 + O(\omega^2 h^2) \tag{2.11d}$$
$$\tilde{c} = c + O(\omega h) \tag{2.11e}$$

This results shows that the numerical wave speed is different then the actual wave speed, with an error depending on the discretization scheme, wavenumber and grid size. The actual error in the amplitude arising from this difference in wave speed is determined by

$$\| \exp^{i\omega n x} - \exp^{i\omega\tilde{n}x} \| = \|1 - \exp^{i\omega(\tilde{n}-n)x} \| \leq C\omega\|\tilde{n} - n\| \leq C\omega^2 h \tag{2.12}$$

This result shows that the error in the solution depends more strongly on the wavenumber then on the grid size. This can best be noticed by determining the best approximation error on the prescribed discrete spaces. Since this error is solely defined by the discrete spaces and therefore incorporates both the discretization error as well as the interpolation error, any deviation is result of the difference between the actual and numerical wave speed.

2.1.3 Fundamental Properties

The Helmholtz equation is ill-posed for wavenumbers corresponding to eigenvalues of the negative Laplacian, these are resonance modes. Either no solution exists in case of non-zero forcing, or an infinite set of solution exists in case no forcing is applied. With additional damping the equation is always well-posed since waves are dissipated over long distance, however this does not always reflect the natural conditions to be modelled. At higher frequencies, the eigenvalues are denser and a given wavenumber is more likely to coincide with an eigenvalue. To summarize, the following properties may be attributed to the Helmholtz equation.

1. The resulting matrix is symmetric but indefinite since positive and negative eigenvalues exists, as shown in section 3.2. Therefore the matrix is slow to be solved by iterative solvers.

2. The problem is ill-conditioned when the wavenumber corresponds to an eigenvalue of the negative laplacian.

3. The pollution error ($O(h^2\omega^3)$) requires large amount of grid points per wavelength to ensure accurate results (10-20 for 2nd degree polynomials and average wavenumbers)

4. Unbounded domains require additional measures (absorbing boundaries, additional Perfectly Matching Layer (PML) or boundary integral representation) to prevent unwanted reflections and resonance modes as discussed in section 3.4.

5. High frequency waves are poorly represented by higher order polynomials therefore other types of trial and test functions are required.

2.2 Mimetic Theory

In this section, the mimetic theory is introduced, for a more extensive overview, the reader is referred to the work by Kreeft [19]. Mimetic theory comprises of two main elements, which are differential geometry and algebraic topology. These concepts describe the physical world just as differential- and vector calculus do. Differential geometry associates variables to geometric shapes by means of integration whereas differential- and vector calculus do not. In this way a distinction is made between metric-free operations, such as differentiation and operations that need metric, such as inner products.

2.2.1 Differential Geometry

Differential geometry is the continuous representation of physical variables in an equation [32]. Variables in differential geometry are called forms, a k-form is an entity that needs to be integrated on a k-dimensional region. So a 0-form is associated with a point, 1-form with a line and 2-form with a surface. These forms can either be inner-oriented or outer-oriented. An outer-oriented 1-form is a variable passing through a line, such as a flux in 2D, while an inner-oriented 1-form is aligned with a line, such as a velocity. The exterior derivative, d, maps a k-form unto a $(k+1)$-form as shown in the de Rham complex in (2.13). The vertical relations represent the Hodge operation, which are a metric dependent mapping of a k-form unto a $n-k$-form.

$$\begin{array}{lccccccc} \text{inner orientation: } \mathbb{R} \rightarrow & \Lambda^{(0)}(\Omega) & \xrightarrow[\nabla]{d} & \Lambda^{(1)}(\Omega) & \xrightarrow[\nabla\times]{d} & \Lambda^{(2)}(\Omega) & \xrightarrow{d} & 0 \\ & \updownarrow \star & & \updownarrow \star & & \updownarrow \star & & \\ \text{outer orientation: } 0 \leftarrow & \Lambda^{(2)}(\Omega) & \xleftarrow[\nabla\cdot]{d} & \Lambda^{(1)}(\Omega) & \xleftarrow[\nabla\times]{d} & \Lambda^{(0)}(\Omega) & \leftarrow & \mathbb{R} \end{array} \tag{2.13}$$

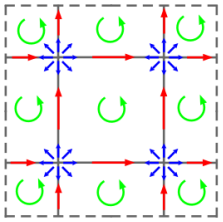


Figure 2.1: Inner oriented cell complex

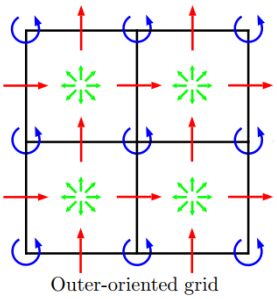


Figure 2.2: Outer oriented cell complex

2.2.2 Algebraic Topology

Algebraic topology is the discrete counterpart of differential geometry. Chains and co-chains are the discrete representation of differential geometry. Fundamental is the cell complex which consists of points, vertices and surfaces as shown in figure 2.1 and figure 2.2, these n-chains correspond to the n-dimensional geometries in differential geometry. The values attached to these chains are the co-chains and are the discrete counterpart of the forms.

When applying this theory to a divergence equation, which is outer-oriented, the metric-free relation becomes clear. The divergence equation is integrated over the corresponding geometry, in this case a surface, S. The divergence term is reduced to a 1 cochain and an algebraic relation remains, relating a 1-cochain to a 2-cochain.

$$\psi + \nabla \cdot \mathbf{u} = 0 \tag{2.14a}$$

$$\int_S (\psi + \nabla \cdot \mathbf{u}) = 0 \tag{2.14b}$$

$$\psi_S + u_N + u_E - u_S - u_W = 0 \tag{2.14c}$$

$$\psi_S + \delta(u_N, u_E, u_S, u_W) = 0 \tag{2.14d}$$

where ψ_S is the 2-form integrated over the corresponding surface and $u_{N,E,S,W}$ the 1-form (flux) integrated over the northern, eastern, southern and western line. The δ-operator in (2.14d) is the coboundary operator which is a mapping of a k-chain unto a $(k+1)$-chain and is the discrete counterpart of the exterior derivative. Dividing the domain into cell complexes allows to create incidence matrices relating the different cochains through their exterior derivative. These incidence matrices are sparse and consist of -1, 0 and 1, representing the metic free and discrete derivative of k-cochains unto $(k+1)$-cochains.

2.2.3 Reduction, Reconstruction and Basis functions

The reduction operator, $\mathcal{R}$, maps a k-form unto a k-cochain. The reduction results in a collection of values at the k-cells. The reduction of the inner 0-form, ϕ, and 1-form $\mathbf{v}$ as well as the outer 1-form, $\mathbf{u}$, and 2-form, ψ, are shown in (2.15) for Lagrange polynomials.

$$\mathcal{R}^0(\phi) = \{\phi_{i,j}\}_{i=0,j=0}^{N,M} \tag{2.15a}$$
$$\mathcal{R}^1(\mathbf{v}) = \{\{u_{i,j}\}_{i=1,j=0}^{N,M}, \{v_{i,j}\}_{i=0,j=1}^{N,M}\} \tag{2.15b}$$
$$\mathcal{R}^1(\mathbf{u}) = \{\{p_{i,j}\}_{i=0,j=1}^{N,M}, \{q_{i,j}\}_{i=1,j=0}^{N,M}\} \tag{2.15c}$$
$$\mathcal{R}^2(\psi) = \{\psi_{i,j}\}_{i=1,j=1}^{N,M} \tag{2.15d}$$

The reconstruction operator, $\mathcal{I}$, introduces the basis functions and reconstruct a continuous differential form from the discrete k-cochain values, this is shown in equation (2.16).

$$\mathcal{I}^0(\phi)(\xi,\eta) = \sum_{i=0}^{N}\sum_{j=0}^{M} \phi_{i,j} h_i(\xi) h_j(\eta) \tag{2.16a}$$
$$\mathcal{I}^1(\mathbf{v})(\xi,\eta) = \sum_{i=1}^{N}\sum_{j=0}^{M} u_{i,j} e_i(\xi) h_j(\eta) + \sum_{i=0}^{N}\sum_{j=1}^{M} v_{i,j} h_i(\xi) e_j(\eta) \tag{2.16b}$$
$$\mathcal{I}^1(\mathbf{u})(\xi,\eta) = \sum_{i=0}^{N}\sum_{j=1}^{M} p_{i,j} h_i(\xi) e_j(\eta) + \sum_{i=1}^{N}\sum_{j=0}^{M} q_{i,j} e_i(\xi) h_j(\eta) \tag{2.16c}$$
$$\mathcal{I}^2(\psi)(\xi,\eta) = \sum_{i=1}^{N}\sum_{j=1}^{M} \psi_{i,j} e_i(\xi) e_j(\eta) \tag{2.16d}$$
$$\tag{2.16e}$$

where h are the Lagrange basis functions as defined by (2.21) and e, the edge basis function derived from the Lagrange functions as shown in (2.17) for $i = 1...N$.

$$e_i(\eta) = -\frac{d}{d\eta}\sum_{k=0}^{i-1} h_k(\eta) \tag{2.17}$$

The reconstruction of a reduction $(\mathcal{I} \circ \mathcal{R})$ equals the projection and is not exact, but an approximation. The reduction of a reconstruction, however is exact.

2.2.4 Mimetic Discretization

Using the ingredients from differential geometry, algebraic topology and the reduction and reconstruction, the term $\nabla\phi$ is discretized as follows. First the 0-form ϕ is reduced to a

collection of 0-cochains living on the vertices of the cell complex. Next, the gradient is applied to the reconstructed term as follows.

$$\nabla\mathcal{I}^0(\phi_{i,j})(\eta,\xi) = \nabla\left(\sum_{i=0}^{N}\sum_{j=0}^{M}\phi_{i,j}h_i(\xi)h_j(\eta)\right) \tag{2.18a}$$

$$= \left(\sum_{i=1}^{N}\sum_{j=0}^{M}(\phi_{i,j}-\phi_{i-1,j})e_i(\xi)h_j(\eta) + \sum_{i=0}^{N}\sum_{j=1}^{M}(\phi_{i,j}-\phi_{i,j-1})h_i(\xi)e_j(\eta)\right) \tag{2.18b}$$

$$= \begin{bmatrix}[e_i(\eta)h_j(\xi)]^T & 0\\ 0 & [h_i(\eta)e_j(\xi)]^T\end{bmatrix} E_{1,0}[\phi_{i,j}] \tag{2.18c}$$

where $E_{1,0}$ is the incidence matrix representing the discrete codifferential relating the 0-cocain to the 1-cochain. (2.18c) shows the matrix representation which is used to create the solvable system of equations.

Similarly, the 1-form, $\mathbf{u}$, can be written in matrix formation as shown in (2.19).

$$\mathbf{u} = \begin{bmatrix}[p_{i,j}]\\ [q_{i,j}]\end{bmatrix}^T \begin{bmatrix}[h_i(\xi)e_j(\eta)] & 0\\ 0 & [e_i(\xi)h_j(\eta)]\end{bmatrix} \tag{2.19}$$

The inner product of $\mathbf{u}$ and $\nabla\phi$ can therefore be written as follows.

$$(\mathbf{u},\nabla\phi) = \left(\begin{bmatrix}[p_{i,j}]\\ [q_{i,j}]\end{bmatrix}^T \begin{bmatrix}[h_i(\xi)e_j(\eta)] & 0\\ 0 & [e_i(\xi)h_j(\eta)]\end{bmatrix}, \begin{bmatrix}[e_i(\eta)h_j(\xi)]^T & 0\\ 0 & [h_i(\eta)e_j(\xi)]^T\end{bmatrix} E_{1,0}[\phi_{i,j}]\right) \tag{2.20a}$$

$$= \begin{bmatrix}[p_{i,j}]\\ [q_{i,j}]\end{bmatrix}^T \begin{bmatrix}\left([h_i(\xi)e_j(\eta)],[e_i(\eta)h_j(\xi)]^T\right) & 0\\ 0 & \left([e_i(\xi)h_j(\eta)],[h_i(\eta)e_j(\xi)]^T\right)\end{bmatrix} E_{1,0}[\phi_{i,j}] \tag{2.20b}$$

$$= \begin{bmatrix}[p_{i,j}]\\ [q_{i,j}]\end{bmatrix}^T M_{\tilde{1},1}E_{1,0}[\phi_{i,j}] \tag{2.20c}$$

where $M_{\tilde{1},1}$ is the mass matrix constituting of all metric dependent relations between the outer-oriented 1-form $\mathbf{u}$ (hence the $\tilde{}$) and the inner oriented 1-form $\nabla\phi$. The inner products are restricted to the basis functions and determined using Gaussian quadrature since Gauss-Lobatto-Legendre nodes are used. Using GLL nodes also reduces the influence of Runge's phenomenon on high order polynomials as compared to evenly spaced grid points.

2.3 Hermite Interpolating Polynomials

2.3.1 Description

In order to mathematically describe the Hermite interpolating polynomials, it is necessary to define the Lagrange basis functions as follows

$$h_i(x) = \prod_{\substack{0 \leq m \leq k \\ m \neq j}} \frac{x - x_m}{x_j - x_m} \tag{2.21}$$

where $0 \leq j \leq k$ and for which the following is true on the domain $[-1\ 1]$:

$$h_i(x_j) = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases} \tag{2.22}$$

The polynomials of the 1st, 2nd and 3rd order are shown on the left in figure 2.3. Using the Lagrange polynomials the Hermite polynomials are defined as follows

$$H_i(x) = (1 - 2 \cdot h_i'(x_i)(x - x_i)) \cdot h_i^2 \tag{2.23}$$
$$H_{i+k+1}(x) = (x - x_i) \cdot h_i^2; \tag{2.24}$$

where $0 \leq i \leq k$ and the following applies.

$$H_i(x_j) = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases} \quad H_i'(x_j) = \begin{cases} 0 & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases} \tag{2.25}$$
$$H_{i+k+1}(x_j) = \begin{cases} 0 & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases} \quad H_{i+k+1}'(x_j) = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases} \tag{2.26}$$

So, Hermite polynomials of the first kind ($k = 1$), consist of a total of 4 polynomials on a single element which either have a value 1 at a boundary or a derivative of 1 at a boundary, while the value and derivatives at the other boundaries are 0. The Hermite polynomials of the 1st and 2nd kind are shown on the right hand side of figure 2.3.

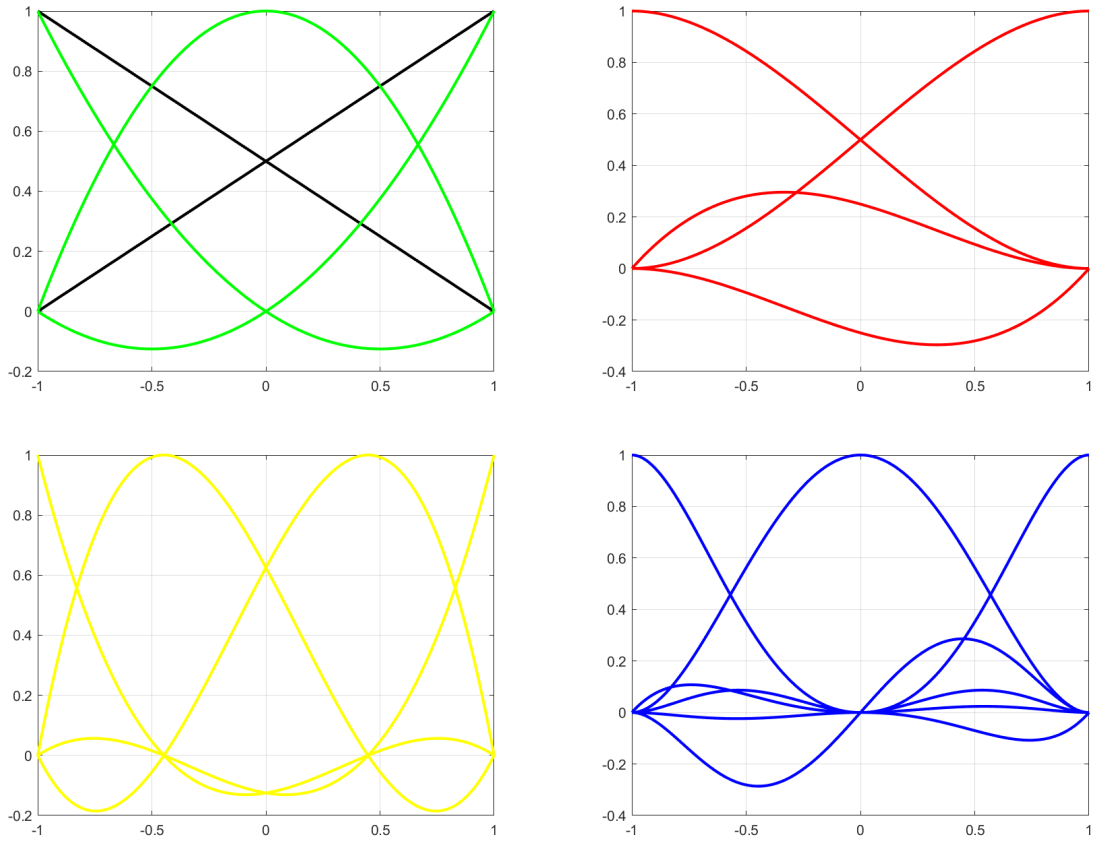


Figure 2.3: Top left: Lagrange Polynomials of the 1st order (black) and 2nd order (green). Bottom left: lagrange Polynomials of the third order (yellow). Top right: Hermite Polynomials of the 1st kind (3rd order, red). Bottom right: Hermite Polynomials of the 2nd kind (5th order, blue).

2.3.2 Interpolating Properties

In order to test the interpolation properties of the proposed polynomials a 2-dimensional smooth function is interpolated, defined by:

$$\phi(X, Y) = \sin(\omega\pi X)\sin(\omega\pi Y) \tag{2.27}$$

where ω is the wavenumber. The functions is interpolated using different polynomial orders for increasing number of elements in both X- and Y-direction. Figure 2.4 shows the interpolation error using Lagrange polynomials ($nD = 0$) and Hermite interpolating polynomials ($nD = 1$). The results show the expected order of convergence of $P_{Order} + 1$.

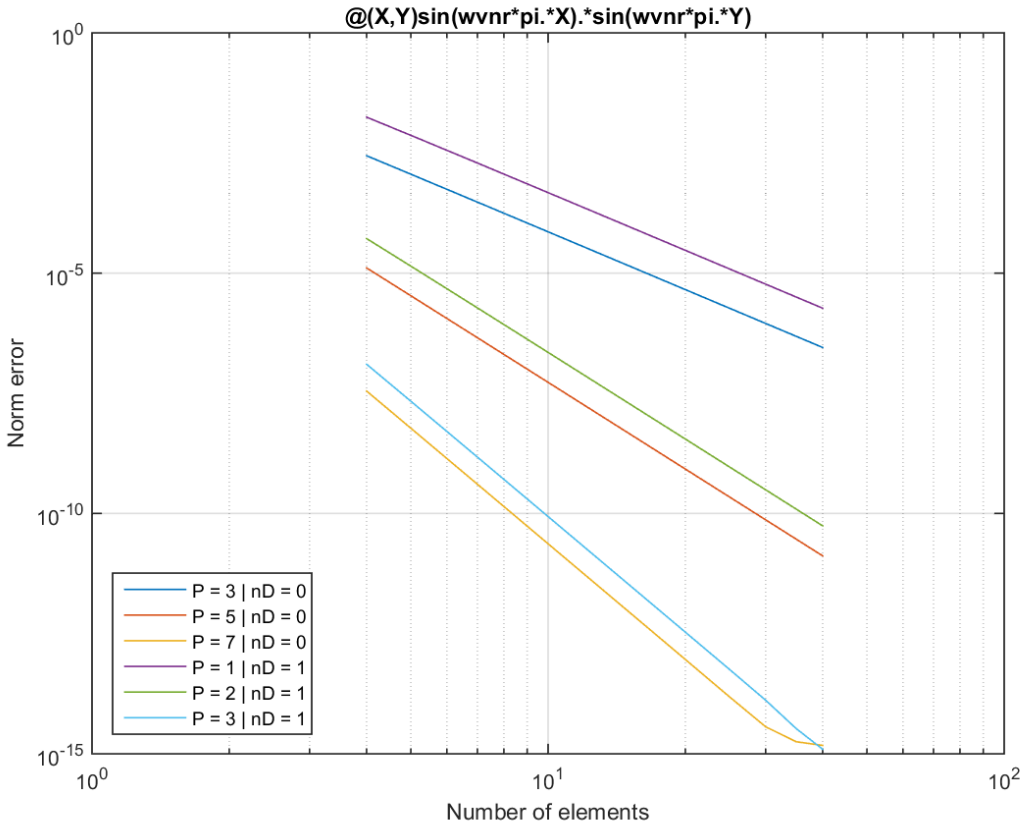


Figure 2.4: Interpolation error for a 2D sin function using nD derivatives and P number of internal elements.

2.3.3 Hermite Interpolating Polynomials applied to mimetic theory

Applying the hermite polynomials to a mimetic discretization in two dimensions results in a set of 49 basis functions making up the 3 main variables. 0-forms are constructed from 16 basis functions, 1-forms from 12 basis function in every direction (32 in total) and 2-forms from 9 basis functions. Figure 2.5 shows the 2-dimensional set of basis function necessary to construct a solution at the lowest order of $P = 1$, equivalent to third order polynomials. The reduction and reconstruction of variables using Hermite polynomials is shown in section 3.3.1 and 3.3.2 respectively.

An overview of the resulting mass matrices are shown in figure 2.6 and figure 2.7, whereas the incidence matrices are shown in figure 2.8 and figure 2.9.

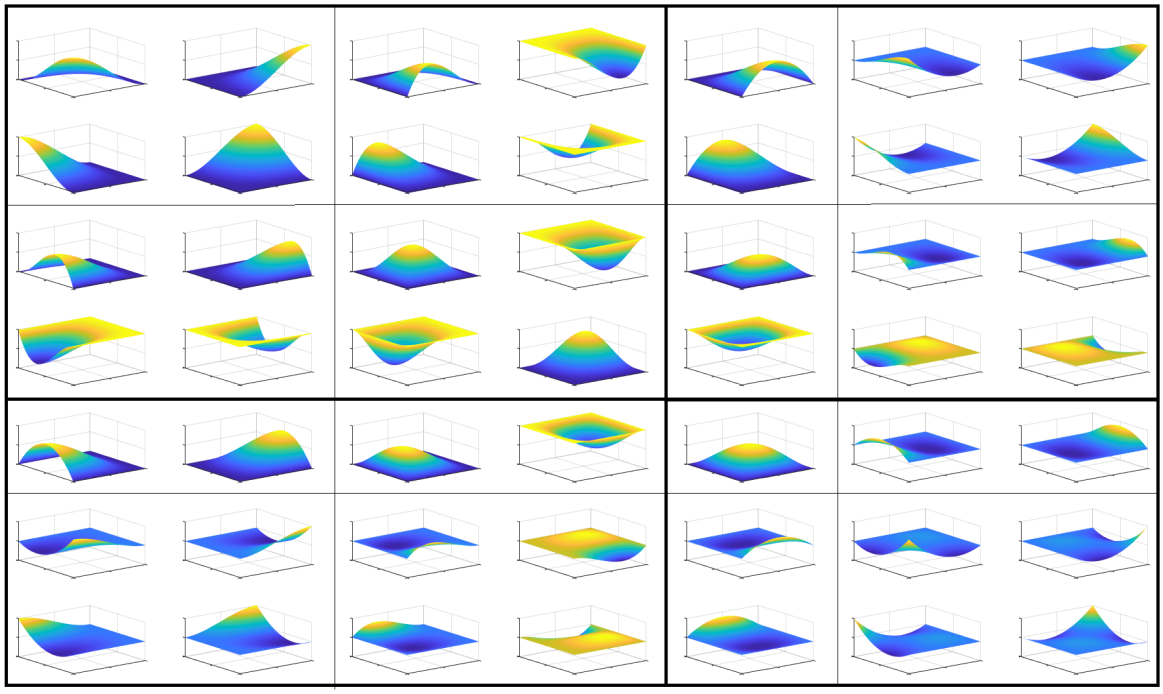


Figure 2.5: Hermite Interpolating Polynomials for P internal elements.

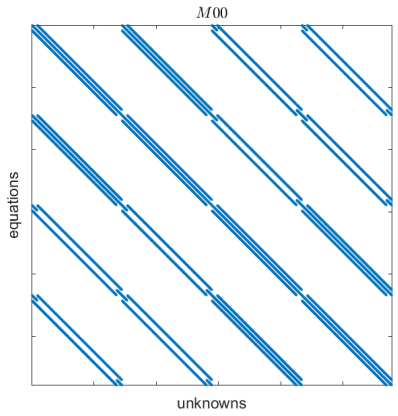


Figure 2.6: M00 mass matrix using Hermite polynomials of the first order and 16 elements.

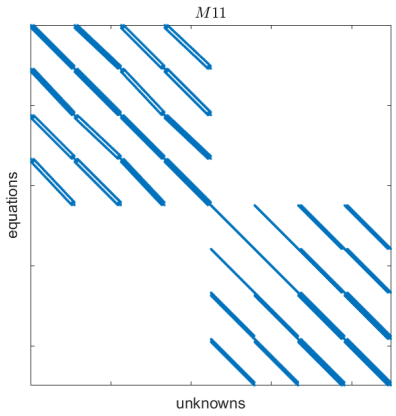


Figure 2.7: M11 mass matrix using Hermite polynomials of the first order and 16 elements.

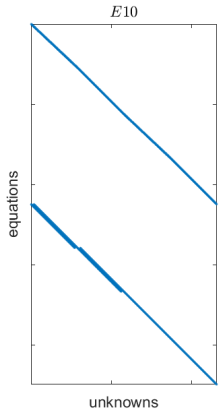


Figure 2.8: E10 incidence matrix using Hermite polynomials of the first order and 16 elements.

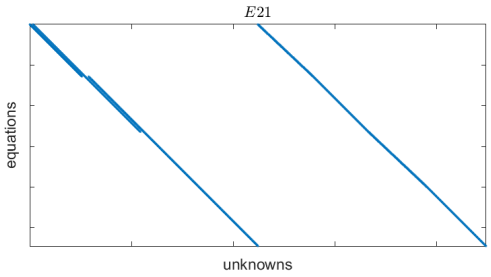


Figure 2.9: E21 incidence matrix using Hermite polynomials of the first order and 16 elements.

2.4 Summary

In this chapter the pollution error has been derived and shown to be fundamental to the most widely applicable discretization methods. The error depends on the mesh size, h, polynomial order, p and wavenumber ,ω as investigated by [22]. Hermite interpolating polynomials are introduced and analysed with respect to convergence rates in 2 dimensions. In accordance with Lagrange polynomials, Hermite polynomials show an interpolation error which reduces by order O+1 as expected. This error is also expected when verifying the implementation of the model presented in the following chapter. When applying hermite polynomials to mimetic theory in 2 dimensions, a set of 49 basis functions is constructed at its lowest order, $P = 1$. In the following chapter the implementation of Hermite polynomials in a least-squares formulation of the Helmholtz equation is presented.

Chapter 3

Helmholtz discretization

The physical representation of variables is of paramount importance to mimetic theory. Therefore the Helmholtz equation will be regarded as model for acoustic waves and will be derived as such in section 3.1. The Helmholtz equation is deconstructed to a 4-variable, 5-equation system of equations and minimized according to the LS principle in section 3.2. The Hermite-derived test functions will then be applied to discretize the functional and define the system of equations. The boundary treatment is discussed and errors are defined. Finally different interpretation of the Helmholtz equation will be discussed in relation to the previous presented model.

3.1 Derivation - Acoustical Disturbances

In order to physically understand the variables involved in the Helmholtz equation, the theory of acoustical disturbances is applied [8]. This is achieved by linearizing the fluid flow equations around a hydrostatic solution and assuming time-harmonic variations. The idea behind this method is to first solve the flow equations to obtain the flow field. A small perturbation determined by linearized equation results in the waves of interest. The Helmholtz equation is obtained by linearizing the isentropic Euler equations as shown in (3.1a) and (3.1b).

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) = 0 \tag{3.1a}$$

$$\frac{\partial \rho \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla)\mathbf{u} + \nabla p = F \tag{3.1b}$$

$$\frac{\partial p}{\partial \rho} = c^2 \tag{3.1c}$$

where ρ is the density, $\mathbf{u}$ the velocity and p the pressure variable. F represents the external

forces and c the isentropic constant relating pressure and density. Assuming zero mean flow and applying the equation of state (3.1c) results in the first order system of equations as shown by (3.2). The pressure and velocity variables have been separated and linearization has been applied to both variables independently, leading to the normalized variables, $p = \frac{p}{\rho_0}$ and $f = \frac{f}{\rho_0}$. A different way is to approach the product of density and velocity as momentum and apply variational calculus to this variable. While the latter interpretation more closely represents the nature of the equations, the former is chosen. Where the variables pressure, density, velocity and flux can be represented by differential k-forms, momentum requires the theory of vector- and covector-valued k-forms. The application of this theory in combination with computational methods is relatively immature and too advanced for this thesis [19].

$$i\omega p + c^2 \nabla \cdot \mathbf{u} = 0 \tag{3.2a}$$
$$\gamma \cdot i\omega \mathbf{u} + \nabla p = f \tag{3.2b}$$

$$\text{where } \gamma = \begin{cases} +1 & \text{Helmholtz equation} \\ -1 & \text{Reaction-Diffusion equation} \end{cases}.$$

$$\Delta p + \gamma \left(\frac{\omega}{c}\right)^2 p = \nabla \cdot f \tag{3.3}$$

The introduction of mimetic theory leads to two different interpretations of (3.2), inner- and outer-oriented, shown respectively in (3.4) and (3.5). Where velocity, $\mathbf{v}^{(1)}$, and flux, $\mathbf{u}^{(n-1)}$ as well as volume- and point-associated pressure, $\psi^{(n)}$ and $\phi^{(0)}$, are each others dual, related by the Hodge star operator, $\star$. (3.6) and (3.7) are the reconstructed single-variable, respectively inner- and outer-oriented, Helmholtz equations.

$$i\omega \phi^{(0)} + c^2 \mathbf{d}^* \mathbf{v}^{(1)} = 0 \tag{3.4a}$$
$$\gamma \cdot i\omega \mathbf{v}^{(1)} + \mathbf{d}\phi^{(0)} = f^{(1)} \tag{3.4b}$$

$$i\omega q^{(n)} + c^2 \mathbf{d}\mathbf{u}^{(n-1)} = 0 \tag{3.5a}$$
$$\gamma \cdot i\omega \mathbf{u}^{(n-1)} + \mathbf{d}^* \psi^{(n)} = f^{(n-1)} \tag{3.5b}$$

$$\mathbf{d}^*\mathbf{d}\phi^{(0)} + \gamma \left(\frac{\omega}{c}\right)^2 \phi^{(0)} = \mathbf{d}^* f^{(1)} \tag{3.6}$$

$$\mathbf{d}\mathbf{d}^*\psi^{(n)} + \gamma \left(\frac{\omega}{c}\right)^2 \psi^{(n)} = \mathbf{d} f^{(n-1)} \tag{3.7}$$

Combining both inner- and outer-oriented relations and rewriting the codifferentials using their dual variables leads to a four field system of equations (3.8). A fifth relation is added to ensure the solution to be curl-free (3.8e). In this setting, equations (3.8a) and (3.8b) are the constitutive relations involving non-exact Hodge-star operators, whereas equations (3.8c), (3.8d) and (3.8e) are metric-free and exact relations. The external forcing is defined as a vector and co-vector as derived from the conservation of momentum relation (3.1b). By defining $f^{(1)} = \mathbf{d}f^{(0)}$ and $f^{(n-1)} = \star\mathbf{d}f^{(0)}$ a point-wise specification of the forcing is allowed.

$$i\omega\phi^{(0)} - c^2 \star \mathbf{d}\mathbf{u}^{(n-1)} = 0 \tag{3.8a}$$
$$\gamma \cdot i\omega\mathbf{u}^{(n-1)} - \star\mathbf{d}\phi^{(0)} = f^{(n-1)} = \star\mathbf{d}f^{(0)} \tag{3.8b}$$
$$\gamma \cdot i\omega\mathbf{v}^{(1)} + \mathbf{d}\phi^{(0)} = \mathbf{d}f^{(0)} \tag{3.8c}$$
$$i\omega\psi^{(n)} + c^2\mathbf{d}\mathbf{u}^{(n-1)} = 0 \tag{3.8d}$$
$$\mathbf{d}\mathbf{v} = 0 \tag{3.8e}$$

The minus sign in (3.8a) and (3.8b) originate from the identity, $\mathbf{d}^* = (-1)^{n(k+1)+1} \star \mathbf{d}\star$.

3.2 Least-Squares problem definition

In choosing a solution method for the constructed system of equations, the indefiniteness of the standard Helmholtz equation formulation has to be taken into account as will be shown for the problem defined by (3.3), with $\frac{w}{c} = k$. An indefinite problem formulation is especially undesirable when solving large problems using iterative solvers as discussed by [12]. In order for the problem to be definite, the variational form of the homogeneous equation must be bound by an appropriate norm of the dependent variable, p. For indefinite problems, an increase in accuracy of the variables representation will not necessarily result in a reduction of the residual, which is undesirable. The variational sesquilinear form, $L(a,b) : H_1 \times H_2 \to \mathbb{C}$, where H_1 and H_2 are respectively trial and test spaces, is defined as follows.

$$\begin{aligned} L(p,\phi) &= (\Delta p, \phi) + \gamma k^2 (p,\phi) \\ &= \langle \nabla p, \phi \rangle - (\nabla p, \nabla \phi) + \gamma k^2 (p,\phi) \\ &= -(\nabla p, \nabla \phi) + \gamma k^2 (p,\phi) \qquad \text{for homogeneous boundary conditions} \end{aligned} \tag{3.9}$$

whereas the k-dependent H^1-norm of the dependent variable p is defined as follows.

$$\|p\|^2_{H^1(\Omega),k} = \|\nabla p\|^2_{L^2} + k^2 \|p\|^2_{L^2} \tag{3.10}$$

For the problem to be definite, the variational form $L(p,p)$ must be bound by the norm 3.10.

$$L(p,p) = \|\nabla p\|_{L^2}^2 - k^2\|p\|_{L^2}^2 \tag{3.11}$$

Due to the different signs, the variational form cannot be bound below by the norm and is therefore indefinite. Furthermore, the eigenvalues of the variational form will be negative when $k^2 > \lambda_1$, where λ_1 is the smallest eigenvalue of the Laplacian and the problem will be ill-defined when $k^2 = \lambda_n$.

As discussed in chapter 1.2.1, multiple methods exists to transform the indefinite Helmholtz equation into a (semi-)definite formulation. For this problem a least-squares discretization is chosen since it provides the most natural way of reconstructing the ideal Rayleigh-Ritz setting of an indefinite problem [17]. The definiteness of the least-squares problem will be discussed further in section 3.2.2 after defining the least-squares problem in 3.2.1.

3.2.1 LS Functional

The least-squares functional and minimization principle associated with (3.8) are defined as follows.

$$\begin{aligned}\mathcal{J}(\phi,\mathbf{u};f) = \frac{1}{2}\Big(\|\frac{1}{c}\left(i\omega\phi - c^2 \star \mathbf{d}\mathbf{u}\right)\|_{X_1}^2 + \|\gamma\cdot i\omega\mathbf{u} - \star\mathbf{d}\phi - f^{(n-1)}\|_{X_2}^2 + \\ \|\gamma\cdot i\omega\mathbf{v} + \mathbf{d}\phi - \mathbf{d}f\|_{X_3}^2 + \|\frac{1}{c}\left(i\omega\psi + c^2\mathbf{d}\mathbf{u}\right)\|_{X_4}^2 + \|\mathbf{d}\mathbf{v}\|_{X_5}^2\Big)\end{aligned} \tag{3.12}$$

$$\min_{(\phi,\mathbf{v})\in U,(\psi,\mathbf{u})\in V} \mathcal{J}((\phi,\mathbf{v}),(\psi,\mathbf{u});f) \tag{3.13}$$

where X_n are the data spaces and U and V the solution spaces. The data spaces are $L^2(\Omega)$ and the solution spaces are defined as $U = H^1(\Omega)\times H(curl,\Omega)$ and $V = (L^2(\Omega))^n \times H(div,\Omega)$, where $\Omega \in \mathbb{C}^n$. As shown in chapter X, the mimetic discretization results in an exact representation of the relations which do not include a Hodge-star operator ($\star$). Therefore excluding these relations from the functional will reduce the number of variables considerably and does not impact the accuracy of the solution. The least-squares problem that will be solved is therefore defined as follows.

$$\mathcal{J}(\phi,\mathbf{u};f) = \frac{1}{2}\Big(\|\frac{1}{c}\left(i\omega\phi - c^2 \star \mathbf{d}\mathbf{u}\right)\|_0^2 + \|\gamma\cdot i\omega\mathbf{u} - \star\mathbf{d}\phi - f^{(n-1)}\|_0^2 + \Big) \tag{3.14}$$

$$\min_{\phi\in U,\mathbf{u}\in V} \mathcal{J}(\phi,\mathbf{u};f) \tag{3.15}$$

where $U = H^1(\Omega)$ and $V = H(div, \Omega)$. Subsequently, the exact relations can be solved for the remaining variables, $\mathbf{v}$ and ψ, by simple algebraic relations.

3.2.2 Norm-equivalence

The norm-equivalence of the functional defined in section 3.2.1, is determined as follows.

$$\underset{p\in U,\mathbf{u}\in V}{\mathcal{J}(p,\mathbf{u};0)} = \|\frac{1}{c}\left(i\omega p + c^2\nabla\cdot\mathbf{u}\right)\|_0^2 + \|\gamma\cdot i\omega\mathbf{u} + \nabla p\|_0^2 \tag{3.16}$$

where $U = H^1(\Omega)$ and $V = H(div, \Omega)$. When expanding the terms in (3.16), applying integration by parts and taking into account the homogeneous boundary conditions, the following result is obtained.

$$\underset{p\in U,\mathbf{u}\in V}{\mathcal{J}(p,\mathbf{u};0)} = \left(\frac{\omega}{c}\right)^2\|p\|_0^2 + \|\nabla p\|_0^2 + \gamma^2\omega^2\|\mathbf{u}\|_0^2 + c^2\|\nabla\cdot\mathbf{u}\|_0^2 + (1+\gamma)\cdot i\omega\left[(\mathbf{u},\nabla p)_0 - (\nabla p,\mathbf{u})_0\right] \tag{3.17a}$$

$$\underset{p\in U,\mathbf{u}\in V}{\mathcal{J}(p,\mathbf{u};0)} = \left(\frac{\omega}{c}\right)^2\|p\|_0^2 + \|\nabla p\|_0^2 + \omega^2\|\mathbf{u}\|_0^2 + c^2\|\nabla\cdot\mathbf{u}\|_0^2 - (1+\gamma)\cdot 2\omega\mathfrak{Im}\left\{(\mathbf{u},\nabla p)_0\right\} \tag{3.17b}$$

The first four terms of the residual (3.17b) can be bound by the appropriate norms of p and $\mathbf{u}$, however, the last inner product cannot be bound by a norm on p or $\mathbf{u}$ and therefore the standard least-squares formulation of the Helmholtz equation is still indefinite. For $\gamma = -1$, the last term vanishes and norm-equivalence is achieved, which is the case for the reaction-diffusion equation. When designing a least-squares method a trade-off must be made between practicality and norm-equivalence. In order to achieve a norm-equivalent definition, fractional norms or negative Sobolov norms are required [17]. Since this project focuses on the discretization of the problem, practicality has been chosen over norm-equivalence.

3.2.3 Minimization of the LS functional

Minimization of the least-squares functional specified by (3.14), according to the principle (3.15) is achieved by applying variational analysis as defined in (3.18).

$$\frac{\partial J(\phi+\epsilon\tilde{\phi},\mathbf{u}+\epsilon\tilde{\mathbf{u}})}{\partial\epsilon} \tag{3.18}$$

The step-by-step minimization is shown in appendix A. The resulting system of equations, after applying integration by parts on the cross-variable terms $\left(\nabla\tilde{\phi}, \mathbf{u}\right)$ and $(\tilde{\mathbf{u}}, \nabla\phi)$, is defined as follows:

$$\begin{aligned} \mathfrak{Re}\left\{k^2\left(\tilde{\phi}, \phi\right) + \left(\star\mathbf{d}\tilde{\phi}, \star\mathbf{d}\phi\right)\right\} - \omega\mathfrak{Im}\left\{(1+\gamma)\cdot\left(\tilde{\phi}, \star\mathbf{du}\right) - \left\langle\star\tilde{\phi}, \mathbf{u}\cdot\mathbf{n}\right\rangle\right\} \\ = \mathfrak{Re}\left\{\left(\star\mathbf{d}\tilde{\phi}, f^{(n-1)}\right)\right\} \\ \mathfrak{Re}\left\{c^2\left(\star\mathbf{d}\tilde{\mathbf{u}}, \star\mathbf{du}\right) + \omega^2\left(\tilde{\mathbf{u}}, \mathbf{u}\right)\right\} + \omega\mathfrak{Im}\left\{(1+\gamma)\cdot\left(\star\mathbf{d}\tilde{\mathbf{u}}, \phi\right) - \left\langle\tilde{\mathbf{u}}\cdot\mathbf{n}, \star\phi\right\rangle\right\} \\ = -\omega\mathfrak{Im}\left\{\left(\tilde{\mathbf{u}}, f^{(n-1)}\right)\right\} \end{aligned} \tag{3.19}$$

where $\mathfrak{Re}\{(a,b)\} = (a_r, b_r) + (a_i, b_i)$, $\mathfrak{Im}\{(a,b)\} = -(a_r, b_i) + (a_i, b_r)$ and $\langle a, b\rangle$ represents the boundary product.

Expanding the system in terms of real and imaginary parts results in an continuous system of 4 equations with 2 scalar and 2 vector unknowns. The next step is to discretize the inner products according to the mimetic method and introduce the Hermite interpolating polynomials as basis functions.

3.3 Discretization

The system as stated in (3.19) consists of inner products which will be discretized using the theory stated in section 2.2. This section will state and discuss the general discretization; the problem-specific characteristics will be discussed in alongside the results in the following chapter.

3.3.1 Reduction, R

The continuous variables ϕ, $\mathbf{v}$, $\mathbf{u}$ and ψ are reduced as follows.

$$R^{0,1}(\phi) = \begin{cases} \phi^{0,0} := \{\phi_{i,j}\}_{i=0,j=0}^{N,M} \\ \phi^{1,0} := \{\left(\frac{\partial^1\phi}{\partial\xi^1}\right)_{i,j}\}_{i=0,j=0}^{N,M} \\ \phi^{0,1} := \{\left(\frac{\partial^1\phi}{\partial\eta^1}\right)_{i,j}\}_{i=0,j=0}^{N,M} \\ \phi^{1,1} := \{\left(\frac{\partial^1\partial^1\phi}{\partial\xi^1\partial\eta^1}\right)_{i,j}\}_{i=0,j=0}^{N,M} \end{cases} \tag{3.20}$$

$$R^{1,1}(\mathbf{v}) = \begin{cases} \{v^{0,0}, u^{0,0}\} := \{v_{i,j}, u_{k,l}\}^{N,M,N,M}_{i=1,j=0,k=0,l=1} \\ \{v^{1,0}, u^{1,0}\} := \{\left(\frac{\partial^1 v}{\partial \xi^1}\right)_{i,j}, \left(\frac{\partial^1 u}{\partial \xi^1}\right)_{k,l}\}^{N,M,N,M}_{i=0,j=0,k=0,l=1} \\ \{v^{0,1}, u^{0,1}\} := \{\left(\frac{\partial^1 v}{\partial \eta^1}\right)_{i,j}, \left(\frac{\partial^1 u}{\partial \eta^1}\right)_{k,l}\}^{N,M,N,M}_{i=1,j=0,k=0,l=0} \\ \{v^{1,1}, u^{1,1}\} := \{\left(\frac{\partial^1 \partial^1 v}{\partial \xi^1 \partial \eta^1}\right)_{i,j}, \left(\frac{\partial^1 \partial^1 u}{\partial \xi^1 \partial \eta^1}\right)_{k,l}\}^{N,M,N,M}_{i=0,j=0,k=0,l=0} \end{cases} \tag{3.21}$$

$$R^{1,1}(\mathbf{u}) = \begin{cases} \{p^{0,0}, q^{0,0}\} := \{p_{i,j}, q_{k,l}\}^{\tilde{N},\tilde{M},\tilde{N},\tilde{M}}_{i=0,j=1,k=1,l=0} \\ \{p^{1,0}, q^{1,0}\} := \{\left(\frac{\partial^1 p}{\partial \xi^1}\right)_{i,j}, \left(\frac{\partial^1 q}{\partial \xi^1}\right)_{k,l}\}^{\tilde{N},\tilde{M},\tilde{N},\tilde{M}}_{i=0,j=1,k=0,l=0} \\ \{p^{0,1}, q^{0,1}\} := \{\left(\frac{\partial^1 p}{\partial \eta^1}\right)_{i,j}, \left(\frac{\partial^1 q}{\partial \eta^1}\right)_{k,l}\}^{\tilde{N},\tilde{M},\tilde{N},\tilde{M}}_{i=0,j=0,k=1,l=0} \\ \{p^{1,1}, q^{1,1}\} := \{\left(\frac{\partial^1 \partial^1 p}{\partial \xi^1 \partial \eta^1}\right)_{i,j}, \left(\frac{\partial^1 \partial^1 q}{\partial \xi^1 \partial \eta^1}\right)_{k,l}\}^{\tilde{N},\tilde{M},\tilde{N},\tilde{M}}_{i=0,j=0,k=0,l=0} \end{cases} \tag{3.22}$$

$$R^{2,1}(\psi) = \begin{cases} \psi^{0,0} := \{\psi_{i,j}\}^{\tilde{N},\tilde{M}}_{i=1,j=1} \\ \psi^{1,0} := \{\left(\frac{\partial^1 \psi}{\partial \xi^1}\right)_{i,j}\}^{\tilde{N},\tilde{M}}_{i=0,j=1} \\ \psi^{0,1} := \{\left(\frac{\partial^1 \psi}{\partial \eta^1}\right)_{i,j}\}^{\tilde{N},\tilde{M}}_{i=1,j=0} \\ \psi^{1,1} := \{\left(\frac{\partial^1 \partial^1 \psi}{\partial \xi^1 \partial \eta^1}\right)_{i,j}\}^{\tilde{N},\tilde{M}}_{i=0,j=0} \end{cases} \tag{3.23}$$

where $R^{n,1}$ is the reduction of the n-form to the 1-derivative of the n-chain. The inner-oriented variables ϕ and $\mathbf{v}$ are discretized on a $N \times M$ grid, while the outer-oriented variables $\mathbf{u}$ and ψ are discretized on a $\tilde{N} \times \tilde{M}$ grid. Reducing a continuous variable results in an approximation error. In this research, reduction up to the first derivative is applied to create a C^1-continuous solution. In short, the reduced values will be defined as $\phi^{d_\xi,d_\eta}_{i,j}$, where d_n and d_m are the derivatives in the respective directions.

3.3.2 Reconstruction, I

The reduced coefficients are recombined into continuous functions as follows.

$$I^0(\phi)(\xi,\eta) = \sum_{d_\xi=0}^{D}\sum_{d_\eta=0}^{D}\sum_{i=0}^{N}\sum_{j=0}^{M} \phi^{d_\xi,d_\eta}_{i,j} h_i^{d_\xi}(\xi) h_j^{d_\eta}(\eta) \tag{3.24}$$

$$\begin{aligned} I^1(\mathbf{v})(\xi,\eta) &= \sum_{i=1}^{N}\sum_{j=0}^{M}\sum_{d_\eta=0}^{D} v^{0,d_\eta}_{i,j} e_i(\xi) h_j^{d_\eta}(\eta) + \sum_{i=0}^{N}\sum_{j=1}^{M}\sum_{d_\xi=0}^{D} u^{d_\xi,0}_{i,j} h_i^{d_\xi}(\xi) e_j(\eta) \\ &+ \sum_{i=0}^{N}\sum_{j=0}^{M}\sum_{d_\xi=1}^{D}\sum_{d_\eta=0}^{D} v^{d_\xi,d_\eta}_{i,j} dh_i^{d_\xi}(\xi) h_j^{d_\eta}(\eta) + \sum_{i=0}^{N}\sum_{j=0}^{M}\sum_{d_\xi=0}^{D}\sum_{d_\eta=1}^{D} u^{d_\xi,d_\eta}_{i,j} h_i^{d_\xi}(\xi) dh_j^{d_\eta}(\eta) \end{aligned} \tag{3.25}$$

$$
\begin{aligned}
I^1(\mathbf{u})(\xi,\eta) &= \sum_{i=0}^{\tilde{N}}\sum_{j=1}^{\tilde{M}}\sum_{d_\xi=0}^{D} p_{i,j}^{d_\xi,0} h_i^{d_\xi}(\xi) e_j(\eta) - \sum_{i=1}^{\tilde{N}}\sum_{j=0}^{\tilde{M}}\sum_{d_\eta=0}^{D} q_{i,j}^{0,d_\eta} e_i(\xi) h_j^{d_\eta}(\eta) \\
&+ \sum_{i=0}^{\tilde{N}}\sum_{j=0}^{\tilde{M}}\sum_{d_\xi=0}^{D}\sum_{d_\eta=1}^{D} p_{i,j}^{d_\xi,d_\eta} h_i^{d_\xi}(\xi) dh_j^{d_\eta}(\eta) - \sum_{i=0}^{\tilde{N}}\sum_{j=0}^{\tilde{M}}\sum_{d_\xi=1}^{D}\sum_{d_\eta=0}^{D} q_{i,j}^{d_\xi,d_\eta} dh_i^{d_\xi}(\xi) h_j^{d_\eta}(\eta)
\end{aligned} \tag{3.26}
$$

$$
\begin{aligned}
I^2(\psi)(\xi,\eta) &= \sum_{i=1}^{\tilde{N}}\sum_{i=1}^{\tilde{M}} \psi_{i,j}^{0,0} e_i(\xi) e_j(\eta) + \sum_{d_\eta=1}^{D}\sum_{i=1}^{\tilde{N}}\sum_{j=0}^{\tilde{M}} \psi_{i,j}^{0,d_\eta} e_i(\xi) dh_j^{d_\eta}(\eta) \\
&+ \sum_{d_\xi=1}^{D}\sum_{i=0}^{\tilde{N}}\sum_{j=1}^{\tilde{M}} \psi_{i,j}^{d_\xi,0} dh_i^{d_\xi}(\xi) e_j(\eta) + \sum_{d_\xi=1}^{D}\sum_{d_\eta=1}^{D}\sum_{i=0}^{\tilde{N}}\sum_{i=0}^{\tilde{M}} \psi_{i,j}^{d_\xi,d_\eta} dh_i^{d_\xi}(\xi) dh_j^{d_\eta}(\eta)
\end{aligned} \tag{3.27}
$$

The corresponding hermite basis function have been discussed in section 2.3. By applying these reconstructed functions to the derived system (3.19), a discrete system of equations is obtained, consisting of mass and incidence matrices. The extensive derivation is shown in Appendix A.

3.4 Boundary Treatment

A Helmholtz domain is either bounded or unbounded. In the case of bounded domains, such as the single wave function in section 4.1, the values at the boundary are known and substituted into the discretized model, the boundaries are therefore enforced. When looking at free wave entering or leaving the domain, a robin boundary condition, known as the Sommerfeld radiation condition, is applied and defined as follows.

$$
-\frac{\partial \phi}{\partial n} + i\omega\phi = i\omega g \tag{3.28a}
$$

$$
i\omega \mathbf{u}\cdot\mathbf{n} + i\omega\phi = i\omega g \tag{3.28b}
$$

$$
\mathbf{u}\cdot\mathbf{n} + \phi = g \tag{3.28c}
$$

where g defines the wave entering the domain and the left hand side allows waves in the domain normal to the boundary to leave without reflection. In this case, however, reflections may occur since only waves in a single direction (normal to the boundary) are allowed to leave the domain. Since The direction is known in the case of a plane wave, this is no source of error. When looking at scattering problems, unwanted reflection may be observed at the open boundaries. Alternatively the boundary may be rewritten as an integral relation to reflect the infinite domain behind the boundary resulting in a additional system of equations. Another

way to deal with an infinite domain is to create an absorbing layer which quickly dampens the waves leaving the domain. However, both methods result in additional computational models as well as resonance modes for certain wavenumbers [29]. In order to simplify the model, the additional reflections are taken for granted when modelling scattering problems. In contrast to enforced boundary conditions in bounded domains, natural boundary conditions are applied to unbounded domains, in which case the prescribed (Robin) boundary condition is substituted to the continuous model and consequently discretized and minimized as well.

3.5 Error definition

In order to establish the accuracy of the resulting model, the following errors will be determined. In the first place the L^2 error as defined by (3.29).

$$L2 = \|\phi - \phi_{ex}\|_{L^2} + \|\mathbf{v} - \mathbf{v}_{ex}\|_{L^2} + \|\mathbf{u} - \mathbf{u}_{ex}\|_{L^2} + \|\psi - \psi_{ex}\|_{L^2} \tag{3.29}$$

Since C^1-continuous function are applied, also the H^1 error is determined as defined by (3.30)

$$\begin{aligned} H1 = \|\phi - \phi_{ex}\|_{L^2} + \|\nabla\phi - \nabla\phi_{ex}\|_{L^2} + \|\mathbf{v} - \mathbf{v}_{ex}\|_{L^2} + \|\nabla\cdot\mathbf{v} - \nabla\cdot\mathbf{v}_{ex}\|_{L^2} + \\ \|\mathbf{u} - \mathbf{u}_{ex}\|_{L^2} + \|\nabla\cdot\mathbf{u} - \nabla\cdot\mathbf{u}_{ex}\|_{L^2} + \|\psi - \psi_{ex}\|_{L^2} + \|\nabla\psi - \nabla\psi_{ex}\|_{L^2} \end{aligned} \tag{3.30}$$

Due to time-limitations it is chosen not to determine and analyse this error, however it is recommended for future research.

The Helmholtz equation is subject to frequency dependent errors, therefore the best approximation error will be determined as well, both L^2 and H^1. Since the best approximation is determined by the interpolation accuracy, the solution is independent of frequency, this will allow for an accurate assessment of the frequency dependent error development.

The best approximation error is determined by minimizing the following functional

$$J = \|\phi - \phi_{ex}\|_{L^2} \tag{3.31}$$

where ϕ is reduced and reconstructed according to section 3.3.1 and 3.3.2 and ϕ_{ex} is reduced using the same trial (basis) functions, but using test functions of higher order. Convergence test showed a minimum order of $2\cdot O(\text{trial})$ was sufficient to determine the best approximation error.

3.6 Finite Element Model

For additional comparison of the results, two similar finite element problems are solved. In the first place a 1-equation finite element model as defined by (3.32)

$$\left(\tilde{\phi}, \Delta\phi\right) + \gamma k^2 \left(\tilde{\phi}, \phi\right) - \left(\tilde{\phi}, \nabla \cdot f\right) = 0 \tag{3.32}$$

Secondly a two-equation finite element model is constructed defined by (3.33)

$$\left(\tilde{\phi}, i\omega\phi\right) + \left(\tilde{\phi}, c^2 \nabla \cdot \mathbf{u}\right) = 0 \tag{3.33a}$$

$$\gamma \cdot i\omega \left(\tilde{\mathbf{u}}, \mathbf{u}\right) + \left(\tilde{\mathbf{u}}, \nabla\phi\right) - \left(\tilde{\mathbf{u}}, f\right) = 0 \tag{3.33b}$$

3.7 Summary

In this chapter the two-equation Helmholtz equation is naturally derived by linearizing the fluid flow equation around a hydrostatic solution and assuming time-harmonic variation. The system of equations is expanded in a four field system by interpreting the two-equation model as inner- and outer oriented. A least-squares minimization definition of the constitutive relations is chosen since it provides a natural way of dealing with indefinite problems. However, it is shown that a definition with $L2$-norms is not norm-equivalent and therefore practicality has been chosen over norm-equivalence. Since the variables are complex-valued, minimizing the least-squares functional results in a system of 4 real valued equations. Both inner- and outer-oriented variables are reduced on the same $N \times M$ grid and reconstructed using the Hermite polynomials. The resulting inner products are defined in terms of mass- and incidence matrices resulting in a full set of equations. The boundaries of unbounded domains are modelled using natural boundary conditions in the form of the Sommerfeld radiation condition, which is a Robin boundary condition. The resulting reflections, in case the outgoing wave direction is unknown, are taken for granted. The bounded domains are modelled using enforced Neumann or Dirichlet boundary conditions. $L2$, $H1$ and best approximation errors are defined where it is chosen not to use the $H1$-errors due to time-limitations. Finally, the problem has been defined using 1- and 2-equation Finite Element formulation for comparison. In the next chapter the code will be verified using a single sinusoidal wave on the reaction diffusion equation ($\gamma = -1$). The pollution error will be analysed by a plane wave experiment and a diffraction and interference problem will be shown.

Chapter 4

Numerical Experiments

In order to establish the performance of the proposed method, several experiments have been conducted. In the first place a single, two dimensional, sinusoidal wave has been modelled to verify the method and determine the convergence properties. Since the main purpose is to verify the implementation of the polynomials as well as the construction of the various mass and incidence matrices, a real valued problem has been chosen. Therefore a different four-field derivation has been chosen to correspond to the domain in $\mathbb{R}^2$. Secondly a plane wave is modelled to analyse the wavenumber-dependence of the errors according to the derivation in $\mathbb{C}^2$. Thirdly a diffraction model has been set-up. The developed code for the single wave in a square domain can be found in Appendix B.2 and the code used to model the diffraction problem is attached in Appendix B.1. Additionally, the functions creating the mass matrices and polynomial basis functions are attached in respectively Appendix B.3 and B.4.

4.1 Single sinusoidal wave

In order to verify the implementation of the hermite polynomials and analyse convergence characteristics, the Helmholtz equation is solved using $\gamma = -1$, which, in this case, corresponds to the reaction-diffusion equation. The equation is decomposed in a four-field system starting from (3.3).

$$\Delta\phi + \gamma\left(\frac{\omega}{c}\right)^2\phi = \nabla\cdot f^{(n-1)} \tag{4.1}$$

The external forcing, f, is obtained from the exact solution and implemented as 0-form, therefore instead of using $\nabla\cdot f^{(n-1)}$, $f^{(0)}$ is used. This leads to the following mimetic four-

field decomposition.

$$\nabla\phi + \mathbf{u} = 0 \tag{4.2a}$$

$$-\nabla\cdot\mathbf{u} + \gamma\left(\frac{\omega}{c}\right)^2\phi = f \tag{4.2b}$$

$$\nabla\phi + \mathbf{v} = 0 \tag{4.2c}$$

$$-\nabla\cdot\mathbf{u} + \psi = 0 \tag{4.2d}$$

with

$$f(x,y) = (-2+\gamma)k^2\sin(kx)\sin(ky) \tag{4.3}$$

where, $k = \left(\frac{\omega}{c}\right)$. On the domain $[0,1]\times[0,1]$, where the exact solution, shown in figure 4.1, is defined as

$$\phi(x,y) = \sin(kx)\sin(ky) \tag{4.4}$$

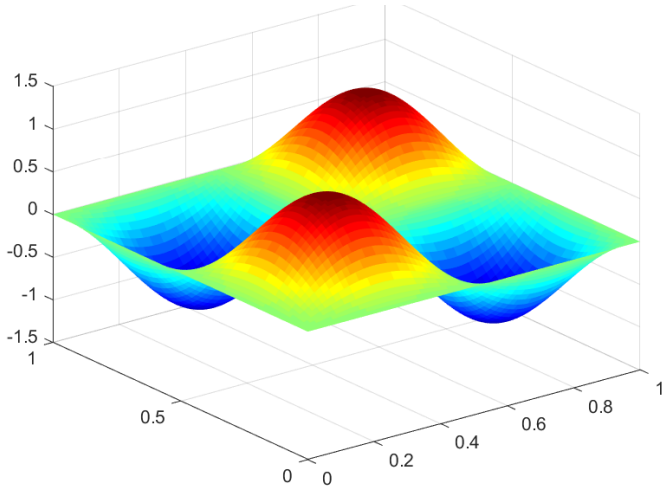


Figure 4.1: Exact solution using $\gamma = 1$ and $k = 2\pi$.

Both inner- and outer oriented variables are solved on the same grid, therefore $\tilde{N} = N$. The system has been solved using second to fifth order ($P = 2...5$) Lagrange polynomials ($nD = 0$) as well as first and second order ($P = 1...2$) Hermite polynomials ($nD = 1$), which correspond to third and fifth order polynomials. The h-dependent L^2-error is shown in figure 4.2 for the various variables. The results show that the order of convergence equals $O(h^{N+1})$ for ϕ and $O(h^N)$ for the variables $\mathbf{v}$, $\mathbf{u}$ and ψ as expected. This result is equivalent to the obtained interpolation convergence rates from section 2.3. Where the constitutive relations (4.2a and 4.2b) are approximated, the fundamental relations (4.2c and 4.2d) are solved up to machine precision as can be seen in graphs **1** and **2** of figure 4.2. The use of Hermite interpolating polynomials does not show a clear distinction in convergence rates. However, when plotting the error as function of the number of unknowns, as shown in figure 4.3, a small distinction can be noticed. The number of unknowns required to solve the problem is less when applying the C^1 continuous polynomials, therefore making the method more cost-efficient as compared to

the use of Lagrange polynomials. However, two downsides arise as well. Firstly, constructing the problem is more complex, in particularly constructing the various matrices. Secondly, the amount of boundary information necessary to accurately calculate the solution is increased, since information about the derivatives is needed as well.

4.2 Plane Wave Experiments

As shown in the previous section, the implementation of Hermite polynomials does not effect the convergence rate of the solution. In this section the Helmholtz equation ($\gamma = 1$) is solved in $\mathbb{C}^2$ using the derivation as defined in section 3.8. The solution will be analyzed for its k-dependence instead of the earlier analyzed h-dependence. This will allow for a conclusion about the pollution error as defined in section 2.1. Figure 4.4 shows the pollution of the solution of a single wave due to the discrepancy between the physical and numerical wave number. Since the wave is defined at the lower left and bottom edges, the phase error is shown to increase in the direction of propagation.

4.2.1 Wave propagation error

In order to investigate this propagation error the following problem is defined.

$$i\omega\phi - c^2 \star \mathbf{d}\mathbf{u} = 0 \tag{4.5a}$$
$$\gamma \cdot i\omega\mathbf{u} - \star\mathbf{d}\phi = f^{(n-1)} = \star\mathbf{d}f^{(0)} \tag{4.5b}$$
$$\gamma \cdot i\omega\mathbf{v} + \mathbf{d}\phi = \mathbf{d}f^{(0)} \tag{4.5c}$$
$$i\omega\psi + c^2\mathbf{d}\mathbf{u} = 0 \tag{4.5d}$$

on $\Omega = [0,1]^2$ and

$$\mathbf{u} \cdot \overrightarrow{\mathbf{n}} - \phi = g \tag{4.6}$$

on $\partial\Omega$, with

$$f(x,y) = k^2(\gamma - 1)\exp^{-ik(x\cos\theta + y\sin\theta)} \tag{4.7a}$$
$$g(x,y) = \phi(x,y)(n_x(x,y)\cos\theta + n_y(x,y)\sin\theta - 1) \tag{4.7b}$$

where $n_x(x,y)$ and $n_y(x,y)$ are the unit normal vector components in x and y direction. The exact solution is defined by

$$\phi(x,y) = \exp^{-ik(x\cos\theta + y\sin\theta)} \tag{4.8}$$

The Robin boundary condition specified in (4.6) is defined as a natural boundary condition and therefore included in the continuous model, discretized and approximated. The variables are expanded as show in section 3.3.2 where the following applies.

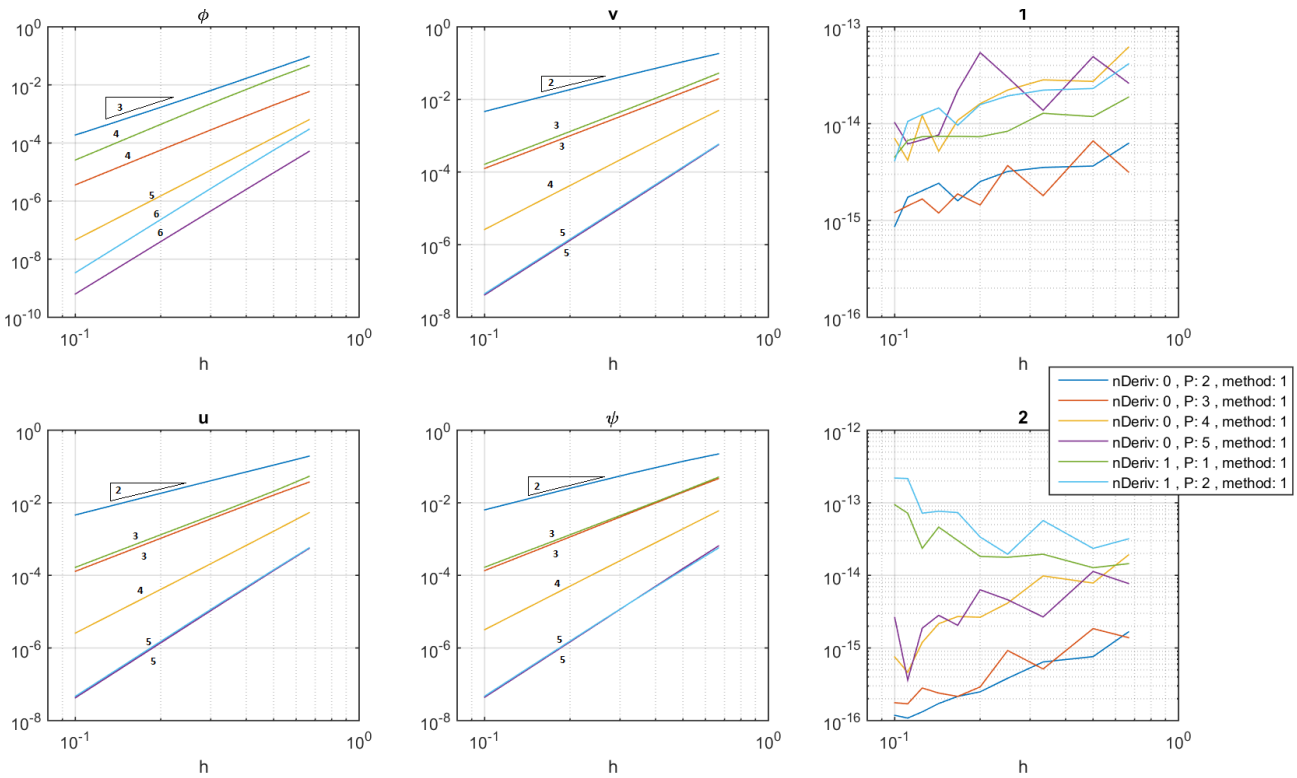


Figure 4.2: h-convergence for the inner-oriented (ϕ and $vvec$) and outer-oriented ($\mathbf{u}$ ψ) variables as well as h-convergence for the fundamental relations $\nabla\phi + \mathbf{u} = 0$ **(1)** and $-\nabla \cdot \mathbf{u} + \psi = 0$ **(2)**. Orders 2 to 5 for the Lagrange polynomials ($nDeriv = 0$) and orders 1 and 2 for the Hermite polynomials ($nDeriv = 0$).

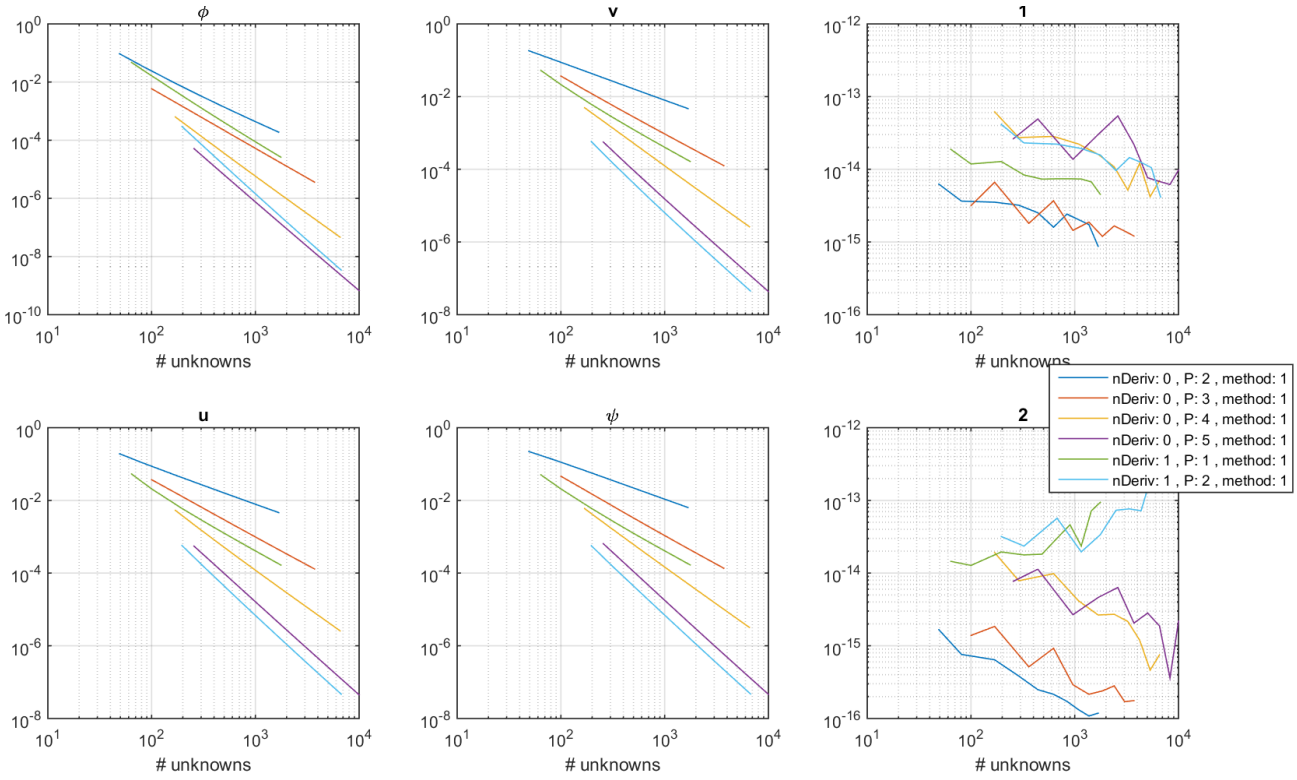


Figure 4.3: L^2 Error as function of number of unknowns for the inner-oriented (ϕ and $vvec$) and outer-oriented ($\mathbf{u}$ ψ) variables as well as errors for the fundamental relations $\nabla\phi + \mathbf{u} = 0$ (**1**) and $-\nabla\cdot\mathbf{u} + \psi = 0$ (**2**). Orders 2 to 5 for the Lagrange polynomials ($nDeriv = 0$) and orders 1 and 2 for the Hermite polynomials ($nDeriv = 0$).

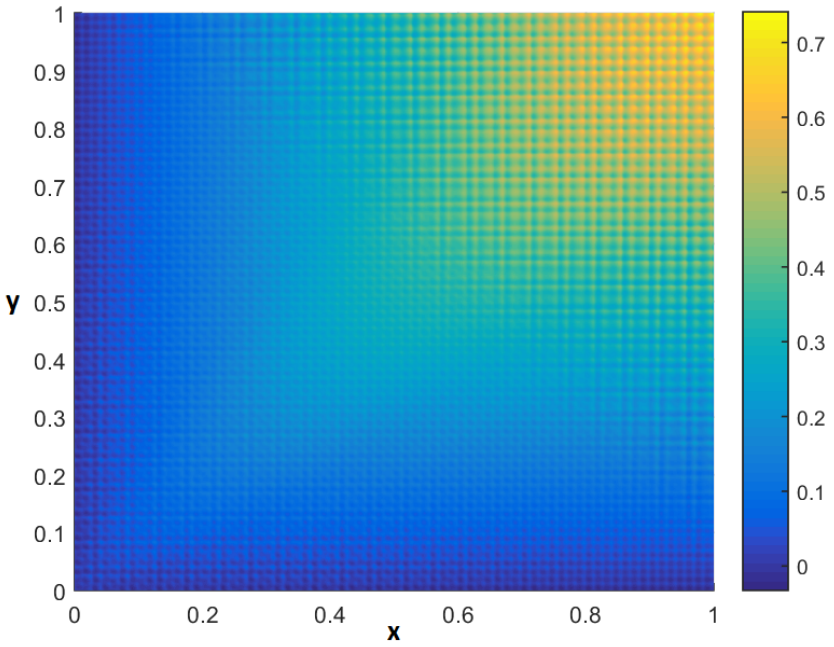


Figure 4.4: The increasing phase error in the direction of wave propagation as defined by $\|\mathfrak{Im}\{\phi\} - \mathfrak{Im}\{\phi_{ex}\}\|$, using $k = 70\pi$, $P = 2$, $nD = 0$ and $\theta = 45\,\text{deg}$

$$\phi^h(\xi,\eta) \in \mathbb{P}^{N,M} \tag{4.9}$$
$$\mathbf{v}^h(\xi,\eta) \in \mathbb{P}^{N-1,M} \times \mathbb{P}^{N,M-1} \tag{4.10}$$
$$\mathbf{u}^h(\xi,\eta) \in \mathbb{P}^{\tilde{N},\tilde{M}-1} \times \mathbb{P}^{\tilde{N}-1,\tilde{M}} \tag{4.11}$$
$$\psi^h(\xi,\eta) \in \mathbb{P}^{\tilde{N}-1,\tilde{M}-1} \times \mathbb{P}^{\tilde{N}-1,\tilde{M}-1} \tag{4.12}$$
$$\tag{4.13}$$

where $N = M = \tilde{N} = \tilde{M}$ is the polynomial degree on the subgrid. The inner- and outer variables are therefore defined on the same grid. The grid is divided into equal elements for which the size h is defined by $h = \frac{1}{k*Nwv}$, where Nwv is the number of elements per wave.

Figure 4.5 shows the relative error of the variables ϕ and $\mathbf{u}$ with respect to the best approximation using the same spaces for a plane wave at 45 degrees using 4 elements per wavelength. As expected, the error increases with increasing wave number for all methods. The results using Hermite polynomials ($nD = 1$) correspond to the result using Lagrange polynomials of the same order (3th and 5th). Figure 4.6 shows the absolute error in ϕ and $\mathbf{u}$ respectively as function of wavenumber. When comparing the two variables, the pollution error is more prominent in ϕ, although the absolute error is smaller for ϕ. The latter observation can be attributed to the higher polynomial order for ϕ (order P for ϕ with respect to order $P-1$ for $\mathbf{u}$).

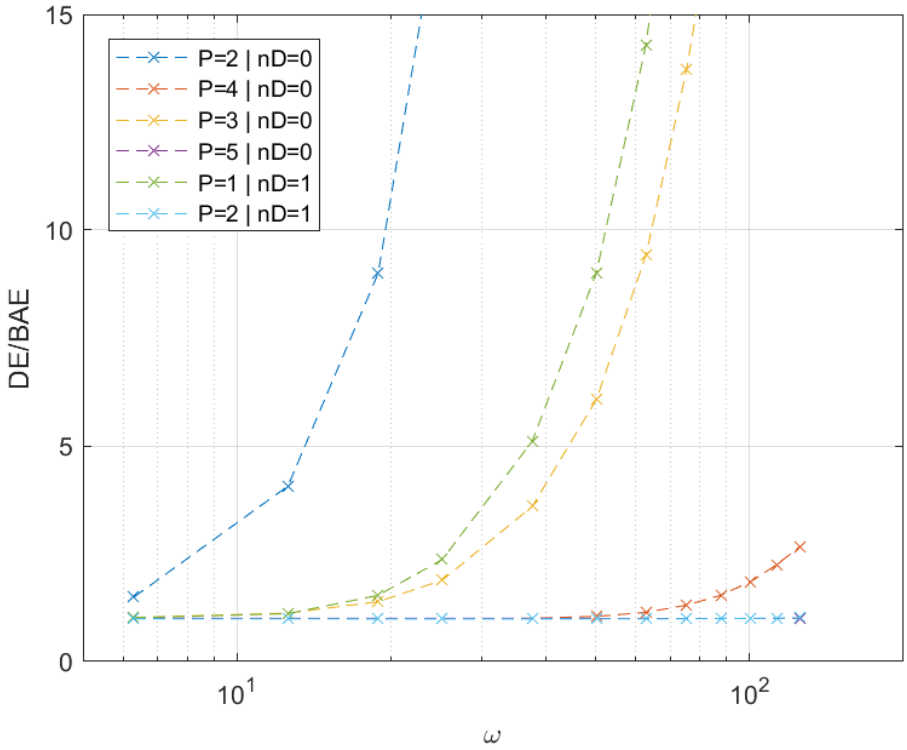


Figure 4.5: Relative error in ϕ and **u** as function of the wavenumber for 4 elements per wavelength at an angele of 45 degrees.

That said, the number of unknowns required is considerably lower when using Hermite polynomials as shown in (4.14) and table 4.1. Where the convergence rates, absolute error and wavenumber dependent error for hermite polynomials of order P corresponds to Lagrange polynomials of order $2P+1$, the number of unknown corresponds to Lagrange polynomials of order $2P$. Therefore, the use of Hermite polynomials is more cost-efficient with respect to lagrange polynomials.

$$nD = 0: \quad n_{nodes} + n_{edges} \tag{4.14a}$$
$$nD = 0: \quad (PN+1)(PM+1) + (PN+1)PM + (PM+1)PN \tag{4.14b}$$
$$nD = 0: \quad 4 \cdot n_{nodes} + (2 \cdot n_{edges} + 4 \cdot n_{nodes}) \tag{4.14c}$$
$$nD = 1: \quad 8 \cdot (PN+1)(PM+1) + 2 \cdot ((PN+1)PM + (PM+1)PN) \tag{4.14d}$$

In order to investigate the convergence even further, figure 4.7 and 4.8 show the same results using only 2 elements per wave. The solution deteriorates for the highest order polynomials at $k = 50$, where quadratic Lagrange elements are unreliable for more then 1 wave in the domain. It can be said that using Hermite polynomials the robustness of the solver with respect to the wave number increases, higher wavenumbers may be calculated accurately using less variables

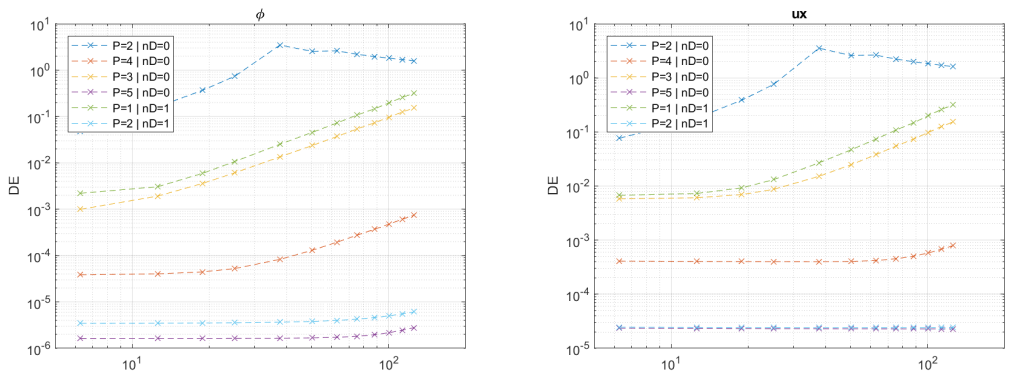


Figure 4.6: Error in ϕ (left) and **u** (right) as function of wavenumber for 4 elements per wavelength at an angle of 45 degrees.

Table 4.1: Number of unknowns for various methods using $N = M = 20$.

	nodes	**edges**	**unknowns**
$nD = 0, P = 2$	1681	3280	4961
$nD = 0, P = 3$	3721	7320	11041
$nD = 0, P = 4$	6561	12960	19521
$nD = 0, P = 5$	10201	20200	30401
$nD = 1, P = 1$	441	840	5208
$nD = 2, P = 2$	1681	3280	20008

as compared to Lagrange polynomials of the same order. However, it does not exceed the performance when using Lagrange polynomials of the same order.

4.2.2 Comparison between 1 and 2 equation FEM models

Figure 4.9 shows the relative error in ϕ with respect to the best approximation for the 1-equation fem model as described by (3.32) and figure 4.10 shows the same error for the 2-equation fem model as described by (3.33). The results show the same deterioration with respect to the wave number, although lower compared to the least-squares formulation. This may be attributed to the lower condition number for the finite element formulation. Since the least-squares variational formulation in its current form does not provide benefits over the Galerkin finite element formulation, it is recommended to either alter the least-squares formulation to a definite form or use the finite element formulation for future research.

4.3 Diffraction

The third and last problem is a characteristic diffraction problem in which a plane wave passes an opening and is diffracted in all directions. The domain has been divided into 6

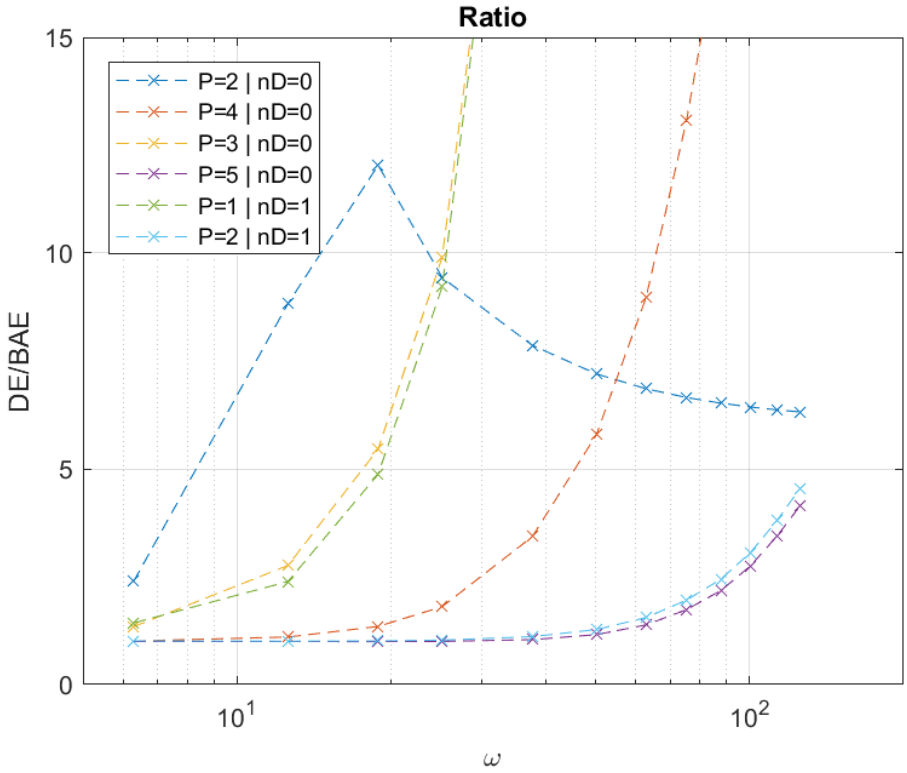


Figure 4.7: Relative error in ϕ and **u** as function of the wavenumber for 2 elements per wavelength at an angele of 45 degrees.

different blocks as shown in figure 4.11. The internal edges between the blocks are merged and boundary conditions are applied. The boundary between blocks 1 and 2 and 5 and 6 are set as walls with

$$\frac{\partial \phi}{\partial n} = 0; \tag{4.15}$$

where n is the unit normal vector pointing outwards (to the right for blocks 1 and 5 and to the left for blocks 2 and 6). This condition ensures all waves are reflected at the wall. The boundaries on the left side are set by the Robin boundary condition.

$$-\frac{\partial \phi}{\partial n} + i\omega\phi = i\omega g; \tag{4.16}$$

where g is defined as an incoming plane wave in horizontal direction. The vertical boundaries at the right side of the domain are to allow waves to leave the domain and to restrict reflections. As discussed in section 3.4, the Robin boundary condition allowing waves to leave in a single direction is chosen due to practicality since no additional system of equations is required to model the infinite domain.

The top and bottom boundaries on the right hand side are modelled in two different ways. In the first place a diffraction model is calculated in which the boundaries are set as an open

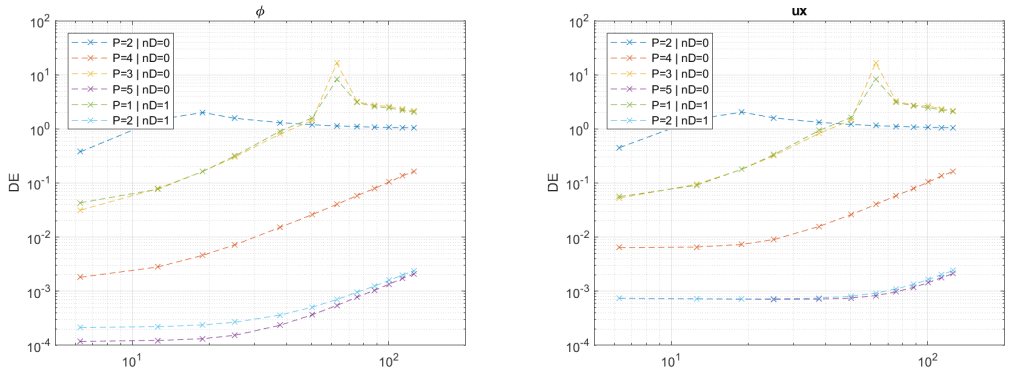


Figure 4.8: Error in ϕ (left) and **u** (right) as function of wavenumber for 2 elements per wavelength at an angle of 45 degrees.

boundary as with the vertical boundary on the right. Secondly an interference model is set-up, for which the top and bottom boundaries are periodic, representing an infinite wall with multiple slits from which diffracted waves interact in the domain.

Figure 4.12 shows the diffracted wave pattern for a wavenumber of 8 and a slit size of three wavelengths. The solution is calculated using 4th order Lagrange polynomials. In order to verify the solution, the minima (or dark fringes) are calculated. Minima are created when waves from one half of the slit cancel out the waves from the other half of the slit. The angles at which these minima occur are determined analytically by

$$w \sin\theta = n\lambda \tag{4.17}$$

where w is the slit size, λ the wavelength, n an integer and θ the angle at which the minima occurs corresponding to the integer. The minima are shown in the figure by the white lines originating from the center of the slit. As can be seen, the analytical minima correspond well with the model.

Since the incoming wave is modelled as a plane wave, an undisturbed plane wave is to be expected on the left hand side of the domain. However, disturbances are visible which may be due to reflections from the unbounded part of the domain as discussed earlier.

Figure 4.13 shows the interference pattern using wavenumber 8 and a slit size of three wavelengths, whereas figure 4.14 shows the interference pattern using a slit size of 1 wavelength. Again reflected waves are visible in the domain due to the limited boundary condition at the free boundaries. Figure 4.15 shows convergence of the solution when increasing the polynomial order from 1 to 5. The solution is plotted at distance 1.5 behind the slit. The solution has visually converged at polynomial order 3.

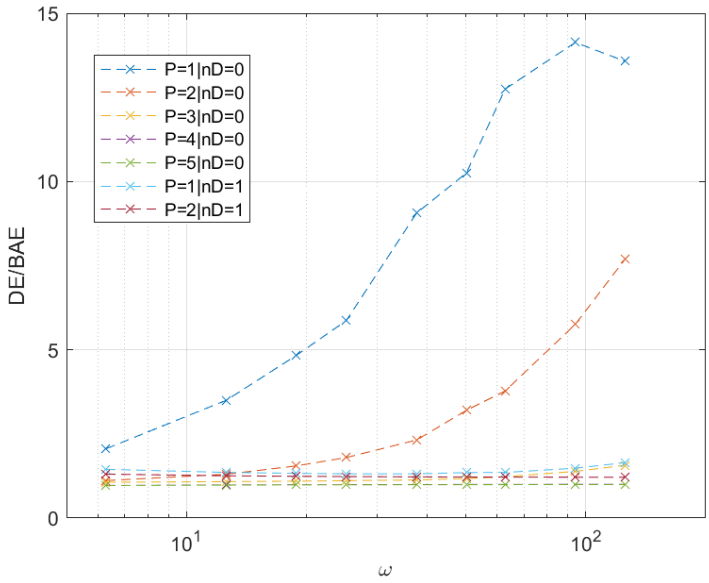


Figure 4.9: Discretization error relative to best approximation as function of wave number for 4 elements per wavelength, theta 45 degrees using a 1-equation finite element method (3.32).

4.4 Summary

The results of the mimetic discretization of the Helmholtz equation with Hermite polynomials have been presented. The model has been verified using a single sinusoidal wave and $\gamma = -1$, which reduced the Helmholtz equation to the sign-definite reaction-diffusion equation. The errors in all variables correspond to the expected interpolation error as determined in section 2.3.2 and the fundamental relations have shown to be solved to machine precision, which justifies, leaving these relations out of the least-squares functional. The use of Hermite polynomials shows no advantage in terms of h-convergence order but is more efficient when related to the number of unknowns. This can also be deduced from the plane wave experiments; applying Hermite polynomials leads to results of polynomial order P at the cost of $P-1$. The same is valid with respect to the pollution error, the polynomial order defines the pollution effect, however the cost for higher polynomial order is lower when using Hermite polynomials. When compared to 1- and 2-equation finite element methods, the finite element formulations show a smaller pollution effect due to the lower condition number. The diffraction model has been modelled using no derivatives. This was not achieved due to time limits. The mimetic discretization using Lagrange polynomials, however shows diffraction patterns as expected. Unwanted reflections occur in the domain due to the simplified Sommerfeld boundary conditions at the unbounded domains. The dark fringes at which the waves are cancelled out correspond well to the theoretical minima. Increasing the order of the model shows convergence of the solution behind the slit which indicates a correct and accurate implementation of the method.

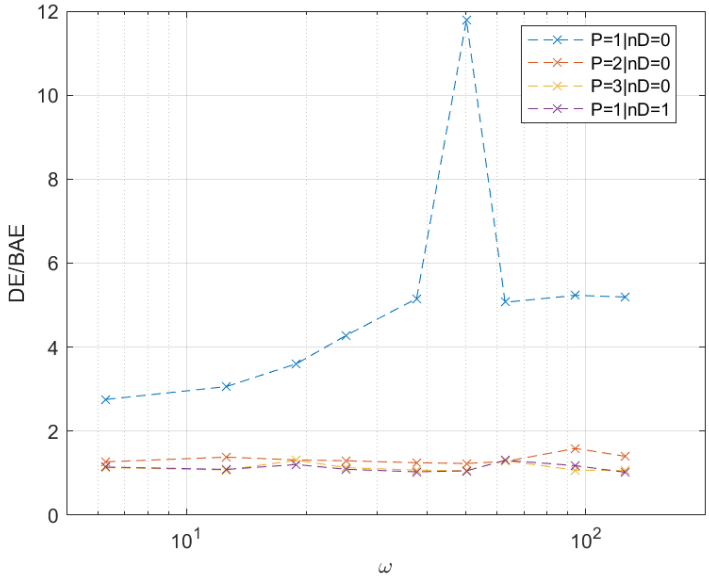


Figure 4.10: Discretization error in ϕ relative to best approximation vs. wave number for 4 elements per wavelength, theta 45 degrees using a 2-equation finite element method (3.33).

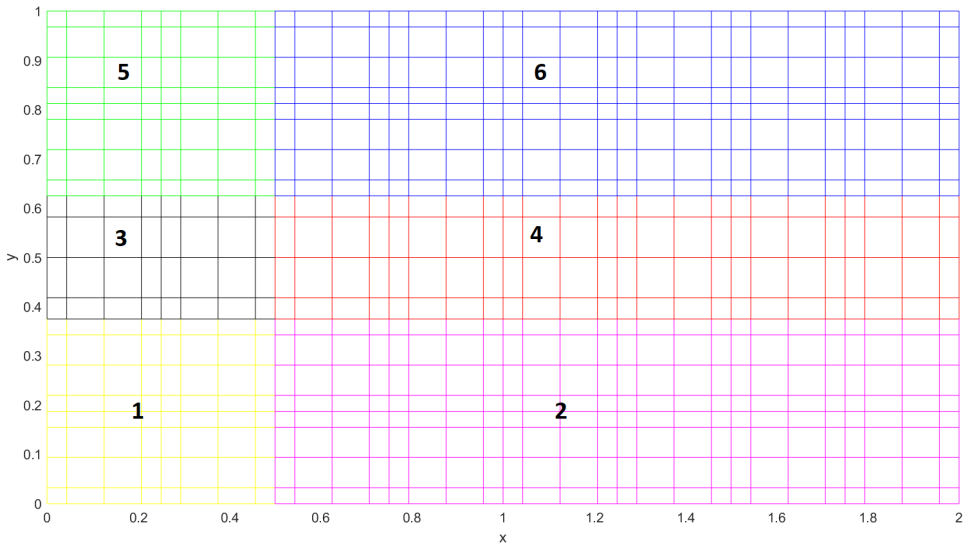


Figure 4.11: Mesh blocks for polynomial order 4, 1 element per wave and wavenumber 4. The boundary between blocks 1 and 2 and 5 and 6 are set to be walls, whereas the boundary between block 3 and 4 represents the slit.

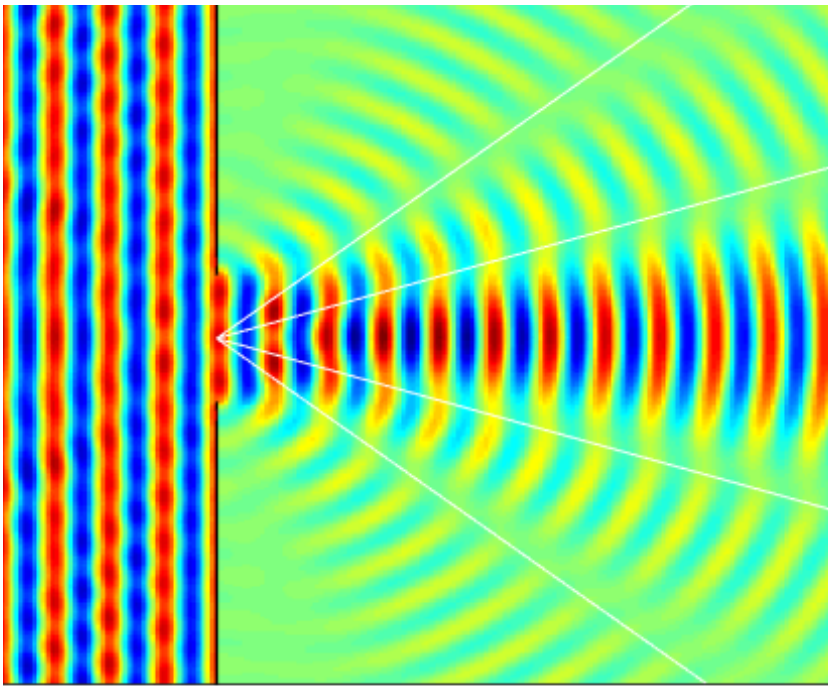


Figure 4.12: Diffraction model for wavenumber 8, a slit size of three wavelengths, polynomial order 4 and using no derivatives.

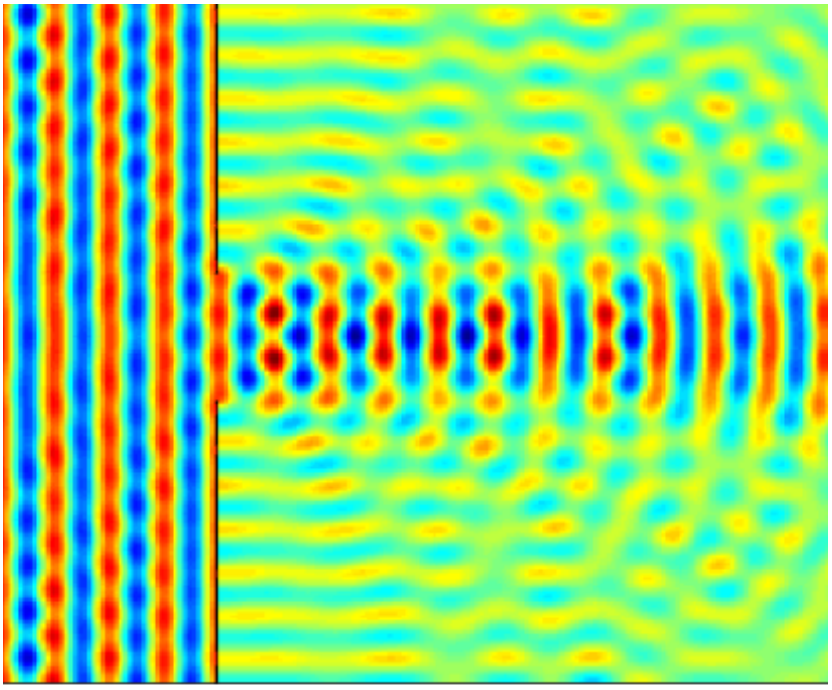


Figure 4.13: Interference model for wavenumber 8, a slit size of three wavelengths, polynomial order 4 and using no derivatives.

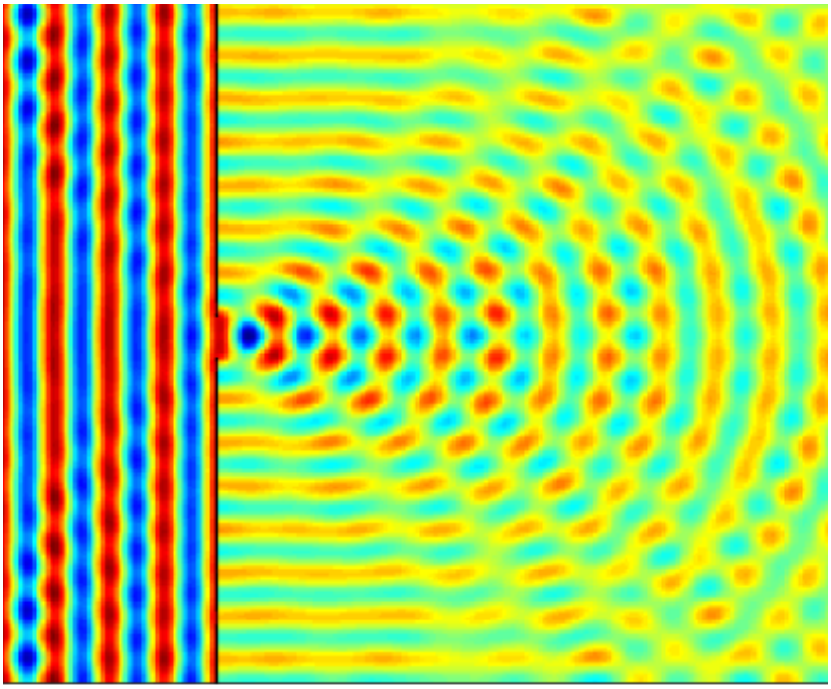


Figure 4.14: Interference model for wavenumber 8, a slit size of one wavelengths, polynomial order 4 and using no derivatives.

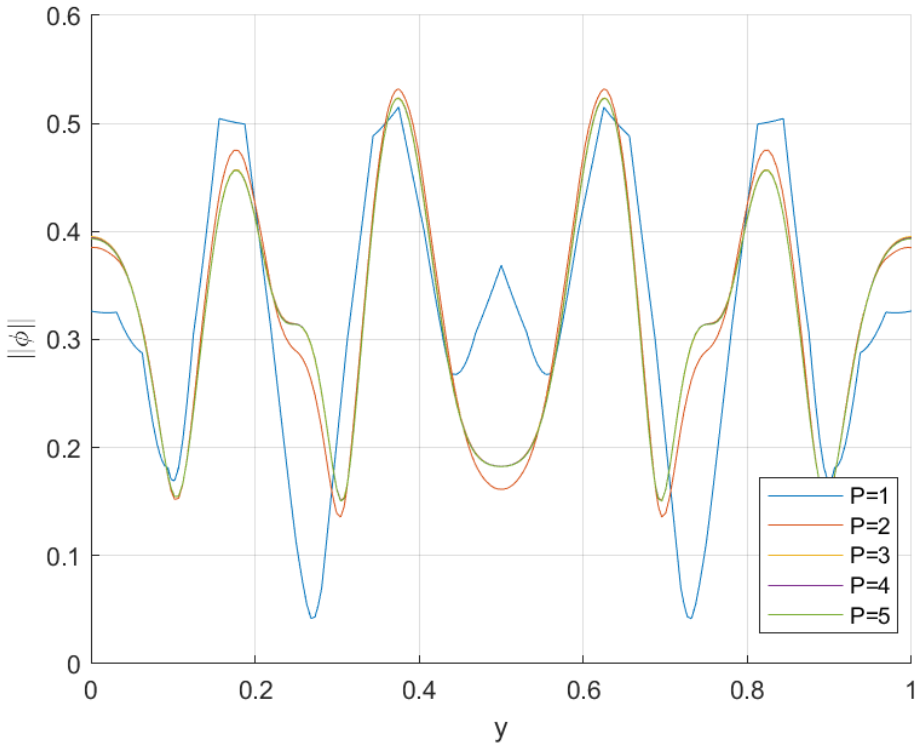


Figure 4.15: Convergence of the solution for orders 1 to 5 at distance 1.5 behind the slit.

Chapter 5

Conclusions and Recommendations

5.1 Conclusions

In this project, a numerical mimetic discretization is applied to model the high frequency Helmholtz equation in two dimensions using Hermite interpolating polynomials. The use of these C^1-continuous polynomials in a 2D mimetic discretization is a novel approach to tackle one of the most challenging problems in scientific computation according to Zienkiewicz [35], i.e. overcoming the pollution error to model high frequency waves in large and complex domains. The main question is therefore to what extent this new approach efficiently and effectively models high frequencies as described by the Helmholtz equation.

The error made constitutes of an discretization error, interpolation error and pollution error. The interpolation error using Hermite polynomials shows the same convergence rate and slightly higher absolute error compared to Lagrange polynomials of the same order. Since the pollution error causes the solution to deteriorate in the direction of wave propagation, this is considered the most important. Applying Hermite interpolating polynomials does not solve the pollution error problem. As discussed by Melenk and Sauter [22], the pollution error for Galerkin methods depends on the mesh size, h polynomial order, p and wavelength, ω. The same conclusion holds for Hermite polynomials. With respect to Lagrange polynomials of order p, Hermite polynomials of the same order show the same deterioration with respect to the wave number. To that extent, the use of hermite polynomials does not model high frequency problems more effectively as compared to Lagrange polynomials. However, the cost at which this result is obtained is lower for Hermite polynomials. Applying Hermite polynomials leads to results of polynomial order P at the cost of $P-1$ when compared to Lagrange polynomials. Therefore the proposed method is more efficient as compared to Lagrange polynomials. This is concluded by only regarding the size of the problem, i.e. number of unknowns. Taking into account the additional time and effort needed for constructing the problem the increased efficiency is negligible.

5.2 Recommendations

Implementing Hermite interpolating polynomials in a 2-dimensional least-squares discretization has been a first step to approach the challenges of the Helmholtz equation. The amount of basis functions and coefficients requires an enormous amount of bookkeeping and perseverance, wherefore several aspects of this projects had to be set aside. Therefore recommendations will follow which provide opportunities to get ahead on this problem.

- In the first place it is recommended to calculate and analyse the $H1$-error. Since the polynomials used are C^1-continuous this will provide more insight in the actual benefits of the method with respect to C^0-continuous polynomials.
- Secondly, but more importantly, it is recommended to analyse the coefficients in terms of their physical interpretation. The key aspect of the mimetic theory is its relation to the physical world. In that light, the physical interpretation of the coefficients sheds light on the relation to the (numerical) wavenumber. A different formulation might ideally result in a wavenumber conservative formulation.
- It is recommended to use a 2-dimensional finite element formulation of the problem. As discussed in section 3.2.2, the standard least squares definition does not provide a norm-equivalent formulation of the problem. Since the main focus of this approach is the use of Hermite polynomials in combination with mimetic discretization, it is recommended to use a more plain variational method as compared to least-squares. As discussed by Babuska and Sauter [3], the pollution error is avoidable in 1 dimension by reformulating the minimization problem, therefore it is recommended to hold on to the 2-dimensional formulation.

Bibliography

[1] I. Babuska and J. M. Melenk. The partition of unity method. *International Journal for Numerical Methods in Engineering*, 40(4):727–758, 1997. URL https://www.scopus.com/inward/record.url?eid=2-s2.0-0031076132&partnerID=40&md5=d14bfb7674a09e8eda8364a7e8a3eca9. cited By 1306.

[2] I. Babuska, F. Ihlenburg, E. T. Paik, and S. A. Sauter. A generalized finite element method for solving the helmholtz equation in two dimensions with minimal pollution. *Computer Methods in Applied Mechanics and Engineering*, 128(3-4):325–359, 1995. doi: 10.1016/0045-7825(95)00890-X. URL http://www.scopus.com/inward/record.url?eid=2-s2.0-0029487468&partnerID=40&md5=58ad97a614ad674b3d466c9b48904e15. cited By 211.

[3] Ivo M. Babuska and Stefan A. Sauter. Is the pollution effect of the fem avoidable for the helmholtz equation considering high wave numbers? *SIAM Journal on Numerical Analysis*, 34(6):2392–2423, 1997. doi: 10.1137/S0036142994269186. URL http://dx.doi.org/10.1137/S0036142994269186.

[4] M. G. Blyth and C. Pozrikidis. A comparative study of the boundary and finite element methods for the helmholtz equation in two dimensions. *Engineering Analysis with Boundary Elements*, 31(1):35–49, 2007. doi: 10.1016/j.enganabound.2006.07.005. URL http://www.scopus.com/inward/record.url?eid=2-s2.0-33845341792&partnerID=40&md5=14cd83d1ebeac9fc9ae70f619f131cdb. cited By 7.

[5] Pavel B. Bochev and Marc I. Gerritsma. A spectral mimetic least-squares method. *Computers and Mathematics with Applications*, 68(11):1480–1502, 2014. doi: 10.1016/j.camwa.2014.09.014. URL http://www.scopus.com/inward/record.url?eid=2-s2.0-84916198475&partnerID=40&md5=d6353ee52b7b30ae22db388c909417fc. cited By 2.

[6] Pavel B. Bochev and James M. Hyman. *Compatible Spatial Discretizations*, chapter Principles of Mimetic Discretizations of Differential Operators, pages 89–119. Springer New York, New York, NY, 2006. ISBN 978-0-387-38034-6. doi: 10.1007/0-387-38034-5_5. URL http://dx.doi.org/10.1007/0-387-38034-5_5.

[7] L. J. Cordova, O. Rojas, B. Otero, and J. Castillo. Compact finite difference modeling of 2-d acoustic wave propagation. *Journal of Computational and Applied Mathematics*, 295:83 – 91, 2016. ISSN 0377-0427. doi: http://dx.doi.org/10.1016/j.cam.2015.01.040. URL

http://www.sciencedirect.com/science/article/pii/S0377042715000618. {VIII} Pan-American Workshop in Applied and Computational Mathematics.

[8] R. Courant and K. O. Friedrichs. *Supersonic Flow and Shock Waves*. Applied Mathematical Sciences. Springer New York, 1948. ISBN 9780387902326. URL https://books.google.nl/books?id=QsxecOQfYw8C.

[9] L. Demkowicz, J. Gopalakrishnan, I. Muga, and J. Zitelli. Wavenumber explicit analysis of a dpg method for the multidimensional helmholtz equation. *Computer Methods in Applied Mechanics and Engineering*, 213-216:126–138, 2012. doi: 10.1016/j.cma.2011.11.024. URL http://www.scopus.com/inward/record.url?eid=2-s2.0-84855802621&partnerID=40&md5=c75960103048e5d89c2df2d4cdaf8c84. cited By 19.

[10] Randall L. Dougherty, Alan S. Edelman, and James M. Hyman. Nonnegativity-, monotonicity-, or convexity-preserving cubic and quintic hermite interpolation. *Mathematics of Computation*, 52(186):471–494, April 1989.

[11] Bjorn Engquist and Olof Runborg. Computational high frequency wave propagation. *Acta Numerica*, 12:181–266, 5 2003. ISSN 1474-0508. doi: 10.1017/S0962492902000119. URL http://journals.cambridge.org/article_S0962492902000119.

[12] O. G. Ernst and M. J. Gander. *Why it is Difficult to Solve Helmholtz Problems with Classical Iterative Methods*, pages 325–363. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012. ISBN 978-3-642-22061-6. doi: 10.1007/978-3-642-22061-6_10. URL http://dx.doi.org/10.1007/978-3-642-22061-6_10.

[13] George J. Fix, Max D. Gunzburger, and R. A. Nicolaides. On finite element methods of the least squares type. *Computers & Mathematics with Applications*, 5(2):87 – 98, 1979. ISSN 0898-1221. doi: http://dx.doi.org/10.1016/0898-1221(79)90062-2. URL http://www.sciencedirect.com/science/article/pii/0898122179900622.

[14] A. Gomez-Polanco, J. M. Guevara-Jordan, and B. Molina. A mimetic iterative scheme for solving biharmonic equations. *Mathematical and Computer Modelling*, 57(9-10):2132–2139, 2013. doi: 10.1016/j.mcm.2011.03.015. URL http://www.scopus.com/inward/record.url?eid=2-s2.0-84875682156&partnerID=40&md5=352cdf7a18ecf391d7afc9ddb41e726f. cited By 2.

[15] D. Gottlieb and S. Orszag. *Numerical Analysis of Spectral Methods*. Society for Industrial and Applied Mathematics, 1977. doi: 10.1137/1.9781611970425. URL http://epubs.siam.org/doi/abs/10.1137/1.9781611970425.

[16] I. G. Graham, M. Lohndorf, J. M. Melenk, and E. A. Spence. When is the error in the h-bem for solving the helmholtz equation bounded independently of k? *BIT Numerical Mathematics*, 55(1):171–214, 2015. doi: 10.1007/s10543-014-0501-5. URL http://www.scopus.com/inward/record.url?eid=2-s2.0-84923677309&partnerID=40&md5=31f2e1e6bc35b5e03046e0ff78bd34ac. cited By 2.

[17] M. D. Gunzburger and P. B. Bochev. *Least-Squares Finite Element Methods*. Springer New York, 2009.

[18] F. Ihlenburg and I. Babuska. Solution of helmholtz problems by knowledge-based fem. *Computer Assisted Mechanics and Engineering Sciences*, 4(3-4):397–415, 1997. URL http://www.scopus.com/inward/record.url?eid=2-s2.0-0031383340&partnerID=40&md5=86dbb13eaa5825519f5404d9144124c1. cited By 9.

[19] J. Kreeft. *Mimetic Spectral Element Method*. Phd thesis, Delft University of Technology, Faculty of Aerospace Engineering, 2013.

[20] B. Lee, T. A. Manteuffel, S. F. Mccormick, and J. Ruge. First-order system least-squares for the helmholtz equation. *SIAM Journal on Scientific Computing*, 21(5):1927–1949, 2000. URL https://www.scopus.com/inward/record.url?eid=2-s2.0-0033693515&partnerID=40&md5=99b1b59e55569fbff47733326bd6b0b3. cited By 34.

[21] S. K. Lele. Compact finite difference schemes with spectral-like resolution. *Journal of Computational Physics*, 103(1):16–42, 1992. doi: 10.1016/0021-9991(92)90324-R. URL http://www.scopus.com/inward/record.url?eid=2-s2.0-9144220381&partnerID=40&md5=9803649817784011944l4df2547c2739. cited By 3159.

[22] J. M. Melenk and S. Sauter. Convergence analysis for finite element discretizations of the helmholtz equation with dirichlet-to-neumann boundary conditions. *Mathematics of Computation*, 79(272):1871–1914, 2010. doi: 10.1090/S0025-5718-10-02362-8. URL http://www.scopus.com/inward/record.url?eid=2-s2.0-77956609030&partnerID=40&md5=af4a3f63d004e5c7430fd4e7785e8a36. cited By 39.

[23] A. Moiola and E. A. Spence. Is the helmholtz equation really sign-indefinite? *SIAM Review*, 56(2):274–312, 2014. doi: 10.1137/120901301. URL http://www.scopus.com/inward/record.url?eid=2-s2.0-84901342997&partnerID=40&md5=5e7e085877f0e4be291373ffed7a24fa. cited By 10.

[24] M. Nabavi, M. H. K. Siddiqui, and J. Dargahi. A new 9-point sixth-order accurate compact finite-difference method for the helmholtz equation. *Journal of Sound and Vibration*, 307(3-5):972–982, 2007. doi: 10.1016/j.jsv.2007.06.070. URL http://www.scopus.com/inward/record.url?eid=2-s2.0-34548569240&partnerID=40&md5=0db7947c21ff4aec26a7fa35d9f5b9e3. cited By 33.

[25] A. Palha, P. P. Rebelo, R. Hiemstra, J. Kreeft, and M. I. Gerritsma. Physics-compatible discretization techniques on single and dual grids, with application to the poisson equation of volume forms. *Journal of Computational Physics*, 257(PB):1394–1422, 2014. doi: 10.1016/j.jcp.2013.08.005. URL http://www.scopus.com/inward/record.url?eid=2-s2.0-84899827364&partnerID=40&md5=99aac5f241483f484ceac9335619aeca. cited By 6.

[26] B. Pluymers, B. Van Hal, D. Vandepitte, and W. Desmet. Trefftz-based methods for time-harmonic acoustics. *Archives of Computational Methods in Engineering*, 14(4):343–381, 2007. doi: 10.1007/s11831-007-9010-x. URL http://www.scopus.com/inward/record.url?eid=2-s2.0-38149084686&partnerID=40&md5=845c497c1484d2e2fd9401858efa43b6. cited By 95.

[27] Jelena Popovic. A fast method for solving the helmholtz equation based on wave splitting. Licenciate thesis, KTH School of Computer Science and Communication, SE-100 44 Stockholm Sweden, 2009.

[28] Jelena Popovic. *Fast Adaptive Numerical Methods for High Frequency Waves and Interface Tracking*. Phd thesis, KTH School of Computer Science and Communication, SE-100 44 Stockholm Sweden, 2012.

[29] Olof Runborg. Helmholtz equation and high frequency approximations. KTH Computer Science and Communication, Spring 2012. DN2255 Numerical Solutions of Differential Equations - Lecture.

[30] Olof Runborg. Computational high frequency wave propagation. KTH Computer Science and Communication, 2007. Isaac Newton Institute - Presentation.

[31] Godehard Sutmann. Compact finite difference schemes of sixth order for the helmholtz equation. *Journal of Computational and Applied Mathematics*, 203(1):15 – 31, 2007. ISSN 0377-0427. doi: http://dx.doi.org/10.1016/j.cam.2006.03.008. URL http://www.sciencedirect.com/science/article/pii/S0377042706001622.

[32] Enzo Tonti. On the mathematical structure of physical theories. *Meccanica*, 7:25–26, 06 1972. doi: 10.1007/BF02128830.

[33] Eli Turkel, Dan Gordon, Rachel Gordon, and Semyon Tsynkov. Compact 2d and 3d sixth order schemes for the helmholtz equation with variable wave number. *Journal of Computational Physics*, 232(1):272 – 287, 2013. ISSN 0021-9991. doi: http://dx.doi.org/10.1016/j.jcp.2012.08.016. URL http://www.sciencedirect.com/science/article/pii/S0021999112004627.

[34] B. Van Genechten, O. Atak, B. Bergen, E. Deckers, S. Jonckheere, J. S. Lee, A. Maressa, K. Vergote, B. Pluymers, D. Vandepitte, and W. Desmet. An efficient wave based method for solving helmholtz problems in three-dimensional bounded domains. *Engineering Analysis with Boundary Elements*, 36(1):63–75, 2012. doi: 10.1016/j.enganabound.2011.07.011. URL https://www.scopus.com/inward/record.url?eid=2-s2.0-80053571942&partnerID=40&md5=7453e03c045bcc828e7e82ef758bd962. cited By 24.

[35] O. C. Zienkiewicz. Achievements and some unsolved problems of the finite element method. *International Journal for Numerical Methods in Engineering*, 47(1-3):9–28, 2000. URL http://www.scopus.com/inward/record.url?eid=2-s2.0-0034628062&partnerID=40&md5=f51a7d3042da7fe61f2db21eb6d5612c. cited By 65.

Appendix A

Discretization

$$-\Delta\phi + \gamma\phi = i\omega f \tag{A.1a}$$
$$\nabla\phi + \mathbf{v} = 0 \tag{A.1b}$$
$$i\omega\mathbf{u} = \mathbf{v} \Rightarrow \nabla\phi + i\omega\mathbf{u} = 0 \tag{A.1c}$$
$$i\omega\nabla\cdot\mathbf{u} + \gamma\phi = i\omega f \Rightarrow \nabla\cdot\mathbf{u} - i\frac{\gamma}{\omega}\phi = f \tag{A.1d}$$
$$\psi + i\frac{\gamma}{\omega}\phi + f = 0 \tag{A.1e}$$
$$\nabla\cdot\mathbf{u} + \psi = 0 \tag{A.1f}$$
$$\tag{A.1g}$$

$$J((\phi,\mathbf{v}),(\psi,\mathbf{u});f) = \frac{1}{2}\Big(\|\nabla\phi + i\omega\mathbf{u}\|_0^2 + \|(\nabla\cdot\mathbf{u} - i\frac{\gamma}{\omega}\phi - f)\|_0^2 + \|\nabla\phi + \mathbf{v}\|_0^2 + \|\nabla\cdot\mathbf{u} + \psi\|_0^2 + \|\nabla\times\mathbf{v}\|_0^2\Big) \tag{A.2a}$$
$$2J(\phi,\mathbf{v};f) = \|\nabla\phi + i\omega\mathbf{u}\|_0^2 + \|\left(\nabla\cdot\mathbf{u} - i\frac{\gamma}{\omega}\phi - f\right)\|_0^2 \tag{A.2b}$$
$$\left.\frac{\partial J}{\partial\epsilon}\right|_{\epsilon=0} = 0, \text{ with } \phi = \phi + \epsilon\tilde{\phi} \text{ and } \mathbf{u} = \mathbf{u} + \epsilon\tilde{\mathbf{u}} \tag{A.2c}$$
$$\phi = \phi_r + i\phi_i \tag{A.2d}$$
$$\mathbf{u} = \mathbf{u}_r + i\mathbf{u}_i \tag{A.2e}$$
$$\epsilon = \epsilon_r + i\epsilon_i \tag{A.2f}$$
$$\frac{\partial}{\partial\epsilon} = \frac{1}{2}\left[\frac{\partial}{\partial\epsilon_r} - i\frac{\partial}{\partial\epsilon_i}\right] \tag{A.2g}$$

$$\begin{aligned}\left(\nabla\tilde{\phi}+i\omega\tilde{\mathbf{u}},\nabla\phi+i\omega\mathbf{u}\right)+\left(\nabla\phi+i\omega\mathbf{u},\nabla\tilde{\phi}+i\omega\tilde{\mathbf{u}}\right)+\\ \left(\nabla\cdot\tilde{\mathbf{u}}-i\frac{\gamma}{\omega}\tilde{\phi},\nabla\cdot\mathbf{u}-i\frac{\gamma}{\omega}\phi-f\right)+\left(\nabla\cdot\mathbf{u}-i\frac{\gamma}{\omega}\phi-f,\nabla\cdot\tilde{\mathbf{u}}-i\frac{\gamma}{\omega}\tilde{\phi}\right)=0\end{aligned} \tag{A.3a}$$

$$\begin{aligned}&\left(\nabla\tilde{\phi},\nabla\phi\right)+\left(\nabla\tilde{\phi},i\omega\mathbf{u}\right)+(i\omega\tilde{\mathbf{u}},\nabla\phi)+(i\omega\tilde{\mathbf{u}},i\omega\mathbf{u})+\\ &\left(\nabla\phi,\nabla\tilde{\phi}\right)+(\nabla\phi,i\omega\tilde{\mathbf{u}})+\left(i\omega\mathbf{u},\nabla\tilde{\phi}\right)+(i\omega\mathbf{u},i\omega\tilde{\mathbf{u}})+\\ &(\nabla\cdot\tilde{\mathbf{u}},\nabla\cdot\mathbf{u})+\left(\nabla\cdot\tilde{\mathbf{u}},-i\frac{\gamma}{\omega}\phi\right)+(\nabla\cdot\tilde{\mathbf{u}},-f)+\\ &\left(-i\frac{\gamma}{\omega}\tilde{\phi},\nabla\cdot\mathbf{u}\right)+\left(-i\frac{\gamma}{\omega}\tilde{\phi},-i\frac{\gamma}{\omega}\phi\right)+\left(-i\frac{\gamma}{\omega}\tilde{\phi},-f\right)+\\ &(\nabla\cdot\mathbf{u},\nabla\cdot\tilde{\mathbf{u}})+\left(\nabla\cdot\mathbf{u},-i\frac{\gamma}{\omega}\tilde{\phi}\right)+\left(-i\frac{\gamma}{\omega}\phi,\nabla\cdot\tilde{\mathbf{u}}\right)+\\ &\left(-i\frac{\gamma}{\omega}\phi,-i\frac{\gamma}{\omega}\tilde{\phi}\right)+(-f,\nabla\cdot\tilde{\mathbf{u}})+\left(-f,-i\frac{\gamma}{\omega}\tilde{\phi}\right)=0\end{aligned} \tag{A.4a}$$

$\tilde{\phi}$-space

$$\begin{aligned}\left(\nabla\tilde{\phi},\nabla\phi\right)+\left(\nabla\phi,\nabla\tilde{\phi}\right)+\left(\nabla\tilde{\phi},i\omega\mathbf{u}\right)+\left(i\omega\mathbf{u},\nabla\tilde{\phi}\right)+\\ \left(-i\frac{\gamma}{\omega}\tilde{\phi},\nabla\cdot\mathbf{u}\right)+\left(\nabla\cdot\mathbf{u},-i\frac{\gamma}{\omega}\tilde{\phi}\right)+\left(-i\frac{\gamma}{\omega}\tilde{\phi},-i\frac{\gamma}{\omega}\phi\right)+\left(-i\frac{\gamma}{\omega}\phi,-i\frac{\gamma}{\omega}\tilde{\phi}\right)\\ =\left(-i\frac{\gamma}{\omega}\tilde{\phi},f\right)+\left(f,-i\frac{\gamma}{\omega}\tilde{\phi}\right)\end{aligned} \tag{A.5a}$$

$$\left(\nabla\tilde{\phi},\nabla\phi\right)+\left(\nabla\phi,\nabla\tilde{\phi}\right)$$

$$\begin{bmatrix}\nabla\tilde{\phi}_r & \nabla\tilde{\phi}_i\end{bmatrix}\begin{bmatrix}1\\ i\end{bmatrix}\begin{bmatrix}1 & -i\end{bmatrix}\begin{bmatrix}\nabla\phi_r\\ \nabla\phi_i\end{bmatrix}+\begin{bmatrix}\nabla\phi_r & \nabla\phi_i\end{bmatrix}\begin{bmatrix}1\\ i\end{bmatrix}\begin{bmatrix}1 & -i\end{bmatrix}\begin{bmatrix}\nabla\tilde{\phi}_r\\ \nabla\tilde{\phi}_i\end{bmatrix} \tag{A.6a}$$

$$\begin{bmatrix}\nabla\tilde{\phi}_r & \nabla\tilde{\phi}_i\end{bmatrix}\begin{bmatrix}1 & -i\\ i & 1\end{bmatrix}\begin{bmatrix}\nabla\phi_r\\ \nabla\phi_i\end{bmatrix}+\begin{bmatrix}\nabla\phi_r & \nabla\phi_i\end{bmatrix}\begin{bmatrix}1 & -i\\ i & 1\end{bmatrix}\begin{bmatrix}\nabla\tilde{\phi}_r\\ \nabla\tilde{\phi}_i\end{bmatrix} \tag{A.6b}$$

$$\begin{bmatrix}\nabla\tilde{\phi}_r & \nabla\tilde{\phi}_i\end{bmatrix}\begin{bmatrix}1 & -i\\ i & 1\end{bmatrix}\begin{bmatrix}\nabla\phi_r\\ \nabla\phi_i\end{bmatrix}+\begin{bmatrix}\nabla\tilde{\phi}_r & \nabla\tilde{\phi}_i\end{bmatrix}\begin{bmatrix}1 & i\\ -i & 1\end{bmatrix}\begin{bmatrix}\nabla\phi_r\\ \nabla\phi_i\end{bmatrix} \tag{A.6c}$$

$$\begin{bmatrix}\nabla\tilde{\phi}_r & \nabla\tilde{\phi}_i\end{bmatrix}\begin{bmatrix}2 & 0\\ 0 & 2\end{bmatrix}\begin{bmatrix}\nabla\phi_r\\ \nabla\phi_i\end{bmatrix}=2\mathfrak{Re}\left\{\left(\nabla\tilde{\phi},\nabla\phi\right)\right\} \tag{A.6d}$$

$$\left(\nabla\tilde{\phi}, i\omega\mathbf{u}\right) + \left(i\omega\mathbf{u}, \nabla\tilde{\phi}\right) = i\omega\left[\left(\mathbf{u}, \nabla\tilde{\phi}\right) - \left(\nabla\tilde{\phi}, \mathbf{u}\right)\right] =$$
$$i\omega\left[\left\langle\mathbf{u}\cdot\mathbf{n}, \tilde{\phi}\right\rangle - \left\langle\tilde{\phi}, \mathbf{u}\cdot\mathbf{n}\right\rangle - \left(\nabla\cdot\mathbf{u}, \tilde{\phi}\right) + \left(\tilde{\phi}, \nabla\cdot\mathbf{u}\right)\right]$$

$$i\omega\left(\begin{bmatrix}\mathbf{u}_r & \mathbf{u}_i\end{bmatrix}\begin{bmatrix}1 & -i\\ i & 1\end{bmatrix}\begin{bmatrix}\nabla\tilde{\phi}_r\\ \nabla\tilde{\phi}_i\end{bmatrix} - \begin{bmatrix}\nabla\tilde{\phi}_r & \nabla\tilde{\phi}_i\end{bmatrix}\begin{bmatrix}1 & -i\\ i & 1\end{bmatrix}\begin{bmatrix}\mathbf{u}_r\\ \mathbf{u}_i\end{bmatrix}\right) \tag{A.7a}$$

$$i\omega\left(\begin{bmatrix}\nabla\tilde{\phi}_r & \nabla\tilde{\phi}_i\end{bmatrix}\begin{bmatrix}1 & i\\ -i & 1\end{bmatrix}\begin{bmatrix}\mathbf{u}_r\\ \mathbf{u}_i\end{bmatrix} - \begin{bmatrix}\nabla\tilde{\phi}_r & \nabla\tilde{\phi}_i\end{bmatrix}\begin{bmatrix}1 & -i\\ i & 1\end{bmatrix}\begin{bmatrix}\mathbf{u}_r\\ \mathbf{u}_i\end{bmatrix}\right) \tag{A.7b}$$

$$i\omega\begin{bmatrix}\nabla\tilde{\phi}_r & \nabla\tilde{\phi}_i\end{bmatrix}\begin{bmatrix}0 & 2i\\ -2i & 0\end{bmatrix}\begin{bmatrix}\mathbf{u}_r\\ \mathbf{u}_i\end{bmatrix} \tag{A.7c}$$

$$\omega\begin{bmatrix}\nabla\tilde{\phi}_r & \nabla\tilde{\phi}_i\end{bmatrix}\begin{bmatrix}0 & -2\\ 2 & 0\end{bmatrix}\begin{bmatrix}\mathbf{u}_r\\ \mathbf{u}_i\end{bmatrix} = 2\omega\mathfrak{Im}\left\{\left(\nabla\tilde{\phi}, \mathbf{u}\right)\right\} = \tag{A.7d}$$

$$2\omega\mathfrak{Im}\left\{\left\langle\tilde{\phi}, \mathbf{u}\cdot\mathbf{n}\right\rangle\right\} - 2\omega\mathfrak{Im}\left\{\left(\tilde{\phi}, \nabla\cdot\mathbf{u}\right)\right\} \tag{A.7e}$$

$$\begin{aligned} i\omega\Bigg(&\begin{bmatrix}\mathbf{u}\cdot\mathbf{n}_r & \mathbf{u}\cdot\mathbf{n}_i\end{bmatrix}\begin{bmatrix}1 & -i\\ i & 1\end{bmatrix}\begin{bmatrix}\tilde{\phi}_r\\ \tilde{\phi}_i\end{bmatrix} - \begin{bmatrix}\tilde{\phi}_r & \tilde{\phi}_i\end{bmatrix}\begin{bmatrix}1 & -i\\ i & 1\end{bmatrix}\begin{bmatrix}\mathbf{u}\cdot\mathbf{n}_r\\ \mathbf{u}\cdot\mathbf{n}_i\end{bmatrix} -\\ &\begin{bmatrix}\nabla\cdot\mathbf{u}_r & \nabla\cdot\mathbf{u}_i\end{bmatrix}\begin{bmatrix}1 & -i\\ i & 1\end{bmatrix}\begin{bmatrix}\tilde{\phi}_r\\ \tilde{\phi}_i\end{bmatrix} + \begin{bmatrix}\tilde{\phi}_r & \tilde{\phi}_i\end{bmatrix}\begin{bmatrix}1 & -i\\ i & 1\end{bmatrix}\begin{bmatrix}\nabla\cdot\mathbf{u}_r\\ \nabla\cdot\mathbf{u}_i\end{bmatrix}\Bigg)\end{aligned} \tag{A.7f}$$

$$\begin{aligned} i\omega\Bigg(&\begin{bmatrix}\tilde{\phi}_r & \tilde{\phi}_i\end{bmatrix}\begin{bmatrix}1 & i\\ -i & 1\end{bmatrix}\begin{bmatrix}\mathbf{u}\cdot\mathbf{n}_r\\ \mathbf{u}\cdot\mathbf{n}_i\end{bmatrix} - \begin{bmatrix}\tilde{\phi}_r & \tilde{\phi}_i\end{bmatrix}\begin{bmatrix}1 & -i\\ i & 1\end{bmatrix}\begin{bmatrix}\mathbf{u}\cdot\mathbf{n}_r\\ \mathbf{u}\cdot\mathbf{n}_i\end{bmatrix} -\\ &\begin{bmatrix}\tilde{\phi}_r & \tilde{\phi}_i\end{bmatrix}\begin{bmatrix}1 & i\\ -i & 1\end{bmatrix}\begin{bmatrix}\nabla\cdot\mathbf{u}_r\\ \nabla\cdot\mathbf{u}_i\end{bmatrix} + \begin{bmatrix}\tilde{\phi}_r & \tilde{\phi}_i\end{bmatrix}\begin{bmatrix}1 & -i\\ i & 1\end{bmatrix}\begin{bmatrix}\nabla\cdot\mathbf{u}_r\\ \nabla\cdot\mathbf{u}_i\end{bmatrix}\Bigg)\end{aligned} \tag{A.7g}$$

$$i\omega\left(\begin{bmatrix}\tilde{\phi}_r & \tilde{\phi}_i\end{bmatrix}\begin{bmatrix}0 & 2i\\ -2i & 0\end{bmatrix}\begin{bmatrix}\mathbf{u}\cdot\mathbf{n}_r\\ \mathbf{u}\cdot\mathbf{n}_i\end{bmatrix} - \begin{bmatrix}\tilde{\phi}_r & \tilde{\phi}_i\end{bmatrix}\begin{bmatrix}0 & 2i\\ -2i & 0\end{bmatrix}\begin{bmatrix}\nabla\cdot\mathbf{u}_r\\ \nabla\cdot\mathbf{u}_i\end{bmatrix}\right) \tag{A.7h}$$

$$\omega\begin{bmatrix}\tilde{\phi}_r & \tilde{\phi}_i\end{bmatrix}\begin{bmatrix}0 & 2\\ -2 & 0\end{bmatrix}\begin{bmatrix}\nabla\cdot\mathbf{u}_r\\ \nabla\cdot\mathbf{u}_i\end{bmatrix} + \omega\begin{bmatrix}\tilde{\phi}_r & \tilde{\phi}_i\end{bmatrix}\begin{bmatrix}0 & -2\\ 2 & 0\end{bmatrix}\begin{bmatrix}\mathbf{u}\cdot\mathbf{n}_r\\ \mathbf{u}\cdot\mathbf{n}_i\end{bmatrix} \tag{A.7i}$$

$\mathbf{u}\cdot\mathbf{n} = i\omega\phi$

$$
\begin{aligned}
&i\omega\Bigg(\begin{bmatrix} i\omega\phi_r & i\omega\phi_i \end{bmatrix} \begin{bmatrix} 1 & -i \\ i & 1 \end{bmatrix} \begin{bmatrix} \tilde{\phi}_r \\ \tilde{\phi}_i \end{bmatrix} - \begin{bmatrix} \tilde{\phi}_r & \tilde{\phi}_i \end{bmatrix} \begin{bmatrix} 1 & -i \\ i & 1 \end{bmatrix} \begin{bmatrix} i\omega\phi_r \\ i\omega\phi_i \end{bmatrix} - \\
&\begin{bmatrix} \nabla\cdot\mathbf{u}_r & \nabla\cdot\mathbf{u}_i \end{bmatrix} \begin{bmatrix} 1 & -i \\ i & 1 \end{bmatrix} \begin{bmatrix} \tilde{\phi}_r \\ \tilde{\phi}_i \end{bmatrix} + \begin{bmatrix} \tilde{\phi}_r & \tilde{\phi}_i \end{bmatrix} \begin{bmatrix} 1 & -i \\ i & 1 \end{bmatrix} \begin{bmatrix} \nabla\cdot\mathbf{u}_r \\ \nabla\cdot\mathbf{u}_i \end{bmatrix} \Bigg)
\end{aligned} \tag{A.7j}
$$

$$
\begin{aligned}
&-\omega^2 \Bigg(\begin{bmatrix} \phi_r & \phi_i \end{bmatrix} \begin{bmatrix} 1 & -i \\ i & 1 \end{bmatrix} \begin{bmatrix} \tilde{\phi}_r \\ \tilde{\phi}_i \end{bmatrix} + \begin{bmatrix} \tilde{\phi}_r & \tilde{\phi}_i \end{bmatrix} \begin{bmatrix} 1 & -i \\ i & 1 \end{bmatrix} \begin{bmatrix} \phi_r \\ \phi_i \end{bmatrix} \Bigg) - \\
&i\omega \Bigg(\begin{bmatrix} \nabla\cdot\mathbf{u}_r & \nabla\cdot\mathbf{u}_i \end{bmatrix} \begin{bmatrix} 1 & -i \\ i & 1 \end{bmatrix} \begin{bmatrix} \tilde{\phi}_r \\ \tilde{\phi}_i \end{bmatrix} - \begin{bmatrix} \tilde{\phi}_r & \tilde{\phi}_i \end{bmatrix} \begin{bmatrix} 1 & -i \\ i & 1 \end{bmatrix} \begin{bmatrix} \nabla\cdot\mathbf{u}_r \\ \nabla\cdot\mathbf{u}_i \end{bmatrix} \Bigg)
\end{aligned} \tag{A.7k}
$$

$$
\begin{aligned}
&-\omega^2 \Bigg(\begin{bmatrix} \tilde{\phi}_r & \tilde{\phi}_i \end{bmatrix} \begin{bmatrix} 1 & i \\ -i & 1 \end{bmatrix} \begin{bmatrix} \phi_r \\ \phi_i \end{bmatrix} + \begin{bmatrix} \tilde{\phi}_r & \tilde{\phi}_i \end{bmatrix} \begin{bmatrix} 1 & -i \\ i & 1 \end{bmatrix} \begin{bmatrix} \phi_r \\ \phi_i \end{bmatrix} \Bigg) - \\
&i\omega \Bigg(\begin{bmatrix} \tilde{\phi}_r & \tilde{\phi}_i \end{bmatrix} \begin{bmatrix} 1 & i \\ -i & 1 \end{bmatrix} \begin{bmatrix} \nabla\cdot\mathbf{u}_r \\ \nabla\cdot\mathbf{u}_i \end{bmatrix} - \begin{bmatrix} \tilde{\phi}_r & \tilde{\phi}_i \end{bmatrix} \begin{bmatrix} 1 & -i \\ i & 1 \end{bmatrix} \begin{bmatrix} \nabla\cdot\mathbf{u}_r \\ \nabla\cdot\mathbf{u}_i \end{bmatrix} \Bigg)
\end{aligned} \tag{A.7l}
$$

$$
-\omega^2 \begin{bmatrix} \tilde{\phi}_r & \tilde{\phi}_i \end{bmatrix} \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix} \begin{bmatrix} \phi_r \\ \phi_i \end{bmatrix} - i\omega \begin{bmatrix} \tilde{\phi}_r & \tilde{\phi}_i \end{bmatrix} \begin{bmatrix} 0 & 2i \\ -2i & 0 \end{bmatrix} \begin{bmatrix} \nabla\cdot\mathbf{u}_r \\ \nabla\cdot\mathbf{u}_i \end{bmatrix} \tag{A.7m}
$$

$$
-\omega^2 \begin{bmatrix} \tilde{\phi}_r & \tilde{\phi}_i \end{bmatrix} \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix} \begin{bmatrix} \phi_r \\ \phi_i \end{bmatrix} + \omega \begin{bmatrix} \tilde{\phi}_r & \tilde{\phi}_i \end{bmatrix} \begin{bmatrix} 0 & 2 \\ -2 & 0 \end{bmatrix} \begin{bmatrix} \nabla\cdot\mathbf{u}_r \\ \nabla\cdot\mathbf{u}_i \end{bmatrix} \tag{A.7n}
$$

$$
\left(-i\tfrac{\gamma}{\omega}\tilde{\phi}, \nabla\cdot\mathbf{u}\right) + \left(\nabla\cdot\mathbf{u}, -i\tfrac{\gamma}{\omega}\tilde{\phi}\right) = i\tfrac{\gamma}{\omega}\left[\left(\nabla\cdot\mathbf{u}, \tilde{\phi}\right) - \left(\tilde{\phi}, \nabla\cdot\mathbf{u}\right)\right]
$$

$$
i\frac{\gamma}{\omega}\left(\begin{bmatrix} \nabla\cdot\mathbf{u}_r & \nabla\cdot\mathbf{u}_i \end{bmatrix} \begin{bmatrix} 1 & -i \\ i & 1 \end{bmatrix} \begin{bmatrix} \tilde{\phi}_r \\ \tilde{\phi}_i \end{bmatrix} - \begin{bmatrix} \tilde{\phi}_r & \tilde{\phi}_i \end{bmatrix} \begin{bmatrix} 1 & -i \\ i & 1 \end{bmatrix} \begin{bmatrix} \nabla\cdot\mathbf{u}_r \\ \nabla\cdot\mathbf{u}_i \end{bmatrix} \right) \tag{A.8a}
$$

$$
i\frac{\gamma}{\omega}\left(\begin{bmatrix} \tilde{\phi}_r & \tilde{\phi}_i \end{bmatrix} \begin{bmatrix} 1 & i \\ -i & 1 \end{bmatrix} \begin{bmatrix} \nabla\cdot\mathbf{u}_r \\ \nabla\cdot\mathbf{u}_i \end{bmatrix} - \begin{bmatrix} \tilde{\phi}_r & \tilde{\phi}_i \end{bmatrix} \begin{bmatrix} 1 & -i \\ i & 1 \end{bmatrix} \begin{bmatrix} \nabla\cdot\mathbf{u}_r \\ \nabla\cdot\mathbf{u}_i \end{bmatrix} \right) \tag{A.8b}
$$

$$
i\frac{\gamma}{\omega} \begin{bmatrix} \tilde{\phi}_r & \tilde{\phi}_i \end{bmatrix} \begin{bmatrix} 0 & 2i \\ -2i & 0 \end{bmatrix} \begin{bmatrix} \nabla\cdot\mathbf{u}_r \\ \nabla\cdot\mathbf{u}_i \end{bmatrix} \tag{A.8c}
$$

$$
\frac{\gamma}{\omega} \begin{bmatrix} \tilde{\phi}_r & \tilde{\phi}_i \end{bmatrix} \begin{bmatrix} 0 & -2 \\ 2 & 0 \end{bmatrix} \begin{bmatrix} \nabla\cdot\mathbf{u}_r \\ \nabla\cdot\mathbf{u}_i \end{bmatrix} \tag{A.8d}
$$

$$
\left(-i\tfrac{\gamma}{\omega}\tilde{\phi}, -i\tfrac{\gamma}{\omega}\phi\right) + \left(-i\tfrac{\gamma}{\omega}\phi, -i\tfrac{\gamma}{\omega}\tilde{\phi}\right) = \left(\tfrac{\gamma}{\omega}\right)^2\left[\left(\tilde{\phi}, \phi\right) + \left(\phi, \tilde{\phi}\right)\right]
$$

$$
\left(\frac{\gamma}{\omega}\right)^2 \left(\begin{bmatrix} \tilde{\phi}_r & \tilde{\phi}_i \end{bmatrix} \begin{bmatrix} 1 & -i \\ i & 1 \end{bmatrix} \begin{bmatrix} \phi_r \\ \phi_i \end{bmatrix} + \begin{bmatrix} \phi_r & \phi_i \end{bmatrix} \begin{bmatrix} 1 & -i \\ i & 1 \end{bmatrix} \begin{bmatrix} \tilde{\phi}_r \\ \tilde{\phi}_i \end{bmatrix} \right) \tag{A.9a}
$$

$$
\left(\frac{\gamma}{\omega}\right)^2 \left(\begin{bmatrix} \tilde{\phi}_r & \tilde{\phi}_i \end{bmatrix} \begin{bmatrix} 1 & -i \\ i & 1 \end{bmatrix} \begin{bmatrix} \phi_r \\ \phi_i \end{bmatrix} + \begin{bmatrix} \tilde{\phi}_r & \tilde{\phi}_i \end{bmatrix} \begin{bmatrix} 1 & i \\ -i & 1 \end{bmatrix} \begin{bmatrix} \phi_r \\ \phi_i \end{bmatrix} \right) \tag{A.9b}
$$

$$
\left(\frac{\gamma}{\omega}\right)^2 \begin{bmatrix} \tilde{\phi}_r & \tilde{\phi}_i \end{bmatrix} \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix} \begin{bmatrix} \phi_r & \phi_i \end{bmatrix} \tag{A.9c}
$$

$$\left(-i\frac{\gamma}{\omega}\tilde{\phi}, f\right) + \left(f, -i\frac{\gamma}{\omega}\tilde{\phi}\right) = i\frac{\gamma}{\omega}\left[\left(f, \tilde{\phi}\right) - \left(\tilde{\phi}, f\right)\right]$$

$$i\frac{\gamma}{\omega}\left(\begin{bmatrix} f_r & f_i \end{bmatrix}\begin{bmatrix} 1 & -i \\ i & 1 \end{bmatrix}\begin{bmatrix} \tilde{\phi}_r \\ \tilde{\phi}_i \end{bmatrix} - \begin{bmatrix} \tilde{\phi}_r & \tilde{\phi}_i \end{bmatrix}\begin{bmatrix} 1 & -i \\ i & 1 \end{bmatrix}\begin{bmatrix} f_r \\ f_i \end{bmatrix}\right) \tag{A.10a}$$

$$i\frac{\gamma}{\omega}\left(\begin{bmatrix} \tilde{\phi}_r & \tilde{\phi}_i \end{bmatrix}\begin{bmatrix} 1 & i \\ -i & 1 \end{bmatrix}\begin{bmatrix} f_r \\ f_i \end{bmatrix} - \begin{bmatrix} \tilde{\phi}_r & \tilde{\phi}_i \end{bmatrix}\begin{bmatrix} 1 & -i \\ i & 1 \end{bmatrix}\begin{bmatrix} f_r \\ f_i \end{bmatrix}\right) \tag{A.10b}$$

$$i\frac{\gamma}{\omega}\begin{bmatrix} \tilde{\phi}_r & \tilde{\phi}_i \end{bmatrix}\begin{bmatrix} 0 & 2i \\ -2i & 0 \end{bmatrix}\begin{bmatrix} f_r \\ f_i \end{bmatrix} \tag{A.10c}$$

$$\frac{\gamma}{\omega}\begin{bmatrix} \tilde{\phi}_r & \tilde{\phi}_i \end{bmatrix}\begin{bmatrix} 0 & -2 \\ 2 & 0 \end{bmatrix}\begin{bmatrix} f_r \\ f_i \end{bmatrix} \tag{A.10d}$$

Continuous System

$$\left(\frac{\gamma}{\omega}\right)^2\left(\tilde{\phi}_r, \phi_r\right) + \left(\nabla\tilde{\phi}_r, \nabla\phi_r\right) - \frac{\gamma}{\omega}\left(\tilde{\phi}_r, \nabla\cdot\mathbf{u}_i\right) - \omega\left(\nabla\tilde{\phi}_r, \mathbf{u}_i\right) = -\frac{\gamma}{\omega}\left(\tilde{\phi}_r, f_i\right) \tag{A.11a}$$

$$\left(\frac{\gamma}{\omega}\right)^2\left(\tilde{\phi}_i, \phi_i\right) + \left(\nabla\tilde{\phi}_i, \nabla\phi_i\right) + \frac{\gamma}{\omega}\left(\tilde{\phi}_i, \nabla\cdot\mathbf{u}_r\right) + \omega\left(\nabla\tilde{\phi}_i, \mathbf{u}_r\right) = \frac{\gamma}{\omega}\left(\tilde{\phi}_i, f_r\right) \tag{A.11b}$$

$\tilde{\mathbf{u}}$-space

$$\begin{aligned}(i\omega\tilde{\mathbf{u}}, \nabla\phi) + (\nabla\phi, i\omega\tilde{\mathbf{u}}) + (i\omega\tilde{\mathbf{u}}, i\omega\mathbf{u}) + (i\omega\mathbf{u}, i\omega\tilde{\mathbf{u}}) + \\ (\nabla\cdot\tilde{\mathbf{u}}, \nabla\cdot\mathbf{u}) + (\nabla\cdot\mathbf{u}, \nabla\cdot\tilde{\mathbf{u}}) + \left(\nabla\cdot\tilde{\mathbf{u}}, -i\frac{\gamma}{\omega}\phi\right) + \left(-i\frac{\gamma}{\omega}\phi, \nabla\cdot\tilde{\mathbf{u}}\right) = \\ (\nabla\cdot\tilde{\mathbf{u}}, f) + (f, \nabla\cdot\tilde{\mathbf{u}})\end{aligned} \tag{A.12a}$$

$$(i\omega\tilde{\mathbf{u}}, \nabla\phi) + (\nabla\phi, i\omega\tilde{\mathbf{u}}) = i\omega\left[(\tilde{\mathbf{u}}, \nabla\phi) - (\nabla\phi, \tilde{\mathbf{u}})\right]$$

$$i\omega\left(\begin{bmatrix} \tilde{\mathbf{u}}_r & \tilde{\mathbf{u}}_i \end{bmatrix}\begin{bmatrix} 1 & -i \\ i & 1 \end{bmatrix}\begin{bmatrix} \nabla\phi_r \\ \nabla\phi_i \end{bmatrix} - \begin{bmatrix} \nabla\phi_r & \nabla\phi_i \end{bmatrix}\begin{bmatrix} 1 & -i \\ i & 1 \end{bmatrix}\begin{bmatrix} \tilde{\mathbf{u}}_r \\ \tilde{\mathbf{u}}_i \end{bmatrix}\right) \tag{A.13a}$$

$$i\omega\left(\begin{bmatrix} \tilde{\mathbf{u}}_r & \tilde{\mathbf{u}}_i \end{bmatrix}\begin{bmatrix} 1 & -i \\ i & 1 \end{bmatrix}\begin{bmatrix} \nabla\phi_r \\ \nabla\phi_i \end{bmatrix} - \begin{bmatrix} \tilde{\mathbf{u}}_r & \tilde{\mathbf{u}}_i \end{bmatrix}\begin{bmatrix} 1 & i \\ -i & 1 \end{bmatrix}\begin{bmatrix} \nabla\phi_r \\ \nabla\phi_i \end{bmatrix}\right) \tag{A.13b}$$

$$i\omega\begin{bmatrix} \tilde{\mathbf{u}}_r & \tilde{\mathbf{u}}_i \end{bmatrix}\begin{bmatrix} 0 & -2i \\ 2i & 0 \end{bmatrix}\begin{bmatrix} \nabla\phi_r \\ \nabla\phi_i \end{bmatrix} \tag{A.13c}$$

$$\omega\begin{bmatrix} \tilde{\mathbf{u}}_r & \tilde{\mathbf{u}}_i \end{bmatrix}\begin{bmatrix} 0 & 2 \\ -2 & 0 \end{bmatrix}\begin{bmatrix} \nabla\phi_r \\ \nabla\phi_i \end{bmatrix} \tag{A.13d}$$

$$(i\omega\tilde{\mathbf{u}}, i\omega\mathbf{u}) + (i\omega\mathbf{u}, i\omega\tilde{\mathbf{u}}) = \omega^2 \left[(\tilde{\mathbf{u}}, \mathbf{u}) + (\mathbf{u}, \tilde{\mathbf{u}})\right]$$

$$\omega^2 \left(\begin{bmatrix} \tilde{\mathbf{u}}_r & \tilde{\mathbf{u}}_i \end{bmatrix} \begin{bmatrix} 1 & -i \\ i & 1 \end{bmatrix} \begin{bmatrix} \mathbf{u}_r \\ \mathbf{u}_i \end{bmatrix} + \begin{bmatrix} \mathbf{u}_r & \mathbf{u}_i \end{bmatrix} \begin{bmatrix} 1 & -i \\ i & 1 \end{bmatrix} \begin{bmatrix} \tilde{\mathbf{u}}_r \\ \tilde{\mathbf{u}}_i \end{bmatrix} \right) \tag{A.14a}$$

$$\omega^2 \left(\begin{bmatrix} \tilde{\mathbf{u}}_r & \tilde{\mathbf{u}}_i \end{bmatrix} \begin{bmatrix} 1 & -i \\ i & 1 \end{bmatrix} \begin{bmatrix} \mathbf{u}_r \\ \mathbf{u}_i \end{bmatrix} + \begin{bmatrix} \tilde{\mathbf{u}}_r & \tilde{\mathbf{u}}_i \end{bmatrix} \begin{bmatrix} 1 & i \\ -i & 1 \end{bmatrix} \begin{bmatrix} \mathbf{u}_r \\ \mathbf{u}_i \end{bmatrix} \right) \tag{A.14b}$$

$$\omega^2 \begin{bmatrix} \tilde{\mathbf{u}}_r & \tilde{\mathbf{u}}_i \end{bmatrix} \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix} \begin{bmatrix} \mathbf{u}_r \\ \mathbf{u}_i \end{bmatrix} \tag{A.14c}$$

$$(\nabla \cdot \tilde{\mathbf{u}}, \nabla \cdot \mathbf{u}) + (\nabla \cdot \mathbf{u}, \nabla \cdot \tilde{\mathbf{u}})$$

$$\begin{bmatrix} \nabla \cdot \tilde{\mathbf{u}}_r & \nabla \cdot \tilde{\mathbf{u}}_i \end{bmatrix} \begin{bmatrix} 1 & -i \\ i & 1 \end{bmatrix} \begin{bmatrix} \nabla \cdot \mathbf{u}_r \\ \nabla \cdot \mathbf{u}_i \end{bmatrix} + \begin{bmatrix} \nabla \cdot \mathbf{u}_r & \nabla \cdot \mathbf{u}_i \end{bmatrix} \begin{bmatrix} 1 & -i \\ i & 1 \end{bmatrix} \begin{bmatrix} \nabla \cdot \tilde{\mathbf{u}}_r \\ \nabla \cdot \tilde{\mathbf{u}}_i \end{bmatrix} \tag{A.15a}$$

$$\begin{bmatrix} \nabla \cdot \tilde{\mathbf{u}}_r & \nabla \cdot \tilde{\mathbf{u}}_i \end{bmatrix} \begin{bmatrix} 1 & -i \\ i & 1 \end{bmatrix} \begin{bmatrix} \nabla \cdot \mathbf{u}_r \\ \nabla \cdot \mathbf{u}_i \end{bmatrix} + \begin{bmatrix} \nabla \cdot \tilde{\mathbf{u}}_r & \nabla \cdot \tilde{\mathbf{u}}_i \end{bmatrix} \begin{bmatrix} 1 & i \\ -i & 1 \end{bmatrix} \begin{bmatrix} \nabla \cdot \mathbf{u}_r \\ \nabla \cdot \mathbf{u}_i \end{bmatrix} \tag{A.15b}$$

$$\begin{bmatrix} \nabla \cdot \tilde{\mathbf{u}}_r & \nabla \cdot \tilde{\mathbf{u}}_i \end{bmatrix} \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix} \begin{bmatrix} \nabla \cdot \mathbf{u}_r \\ \nabla \cdot \mathbf{u}_i \end{bmatrix} \tag{A.15c}$$

$$\left(\nabla \cdot \tilde{\mathbf{u}}, -i\tfrac{\gamma}{\omega}\phi\right) + \left(-i\tfrac{\gamma}{\omega}\phi, \nabla \cdot \tilde{\mathbf{u}}\right) = i\tfrac{\gamma}{\omega}\left[(\nabla \cdot \tilde{\mathbf{u}}, \phi) - (\phi, \nabla \cdot \tilde{\mathbf{u}})\right]$$

$$i\frac{\gamma}{\omega} \left(\begin{bmatrix} \nabla \cdot \tilde{\mathbf{u}}_r & \nabla \cdot \tilde{\mathbf{u}}_i \end{bmatrix} \begin{bmatrix} 1 & -i \\ i & 1 \end{bmatrix} \begin{bmatrix} \phi_r \\ \phi_i \end{bmatrix} - \begin{bmatrix} \phi_r & \phi_i \end{bmatrix} \begin{bmatrix} 1 & -i \\ i & 1 \end{bmatrix} \begin{bmatrix} \nabla \cdot \tilde{\mathbf{u}}_r \\ \nabla \cdot \tilde{\mathbf{u}}_i \end{bmatrix} \right) \tag{A.16a}$$

$$i\frac{\gamma}{\omega} \left(\begin{bmatrix} \nabla \cdot \tilde{\mathbf{u}}_r & \nabla \cdot \tilde{\mathbf{u}}_i \end{bmatrix} \begin{bmatrix} 1 & -i \\ i & 1 \end{bmatrix} \begin{bmatrix} \phi_r \\ \phi_i \end{bmatrix} - \begin{bmatrix} \nabla \cdot \tilde{\mathbf{u}}_r & \nabla \cdot \tilde{\mathbf{u}}_i \end{bmatrix} \begin{bmatrix} 1 & i \\ -i & 1 \end{bmatrix} \begin{bmatrix} \phi_r \\ \phi_i \end{bmatrix} \right) \tag{A.16b}$$

$$i\frac{\gamma}{\omega} \begin{bmatrix} \nabla \cdot \tilde{\mathbf{u}}_r & \nabla \cdot \tilde{\mathbf{u}}_i \end{bmatrix} \begin{bmatrix} 0 & -2i \\ 2i & 0 \end{bmatrix} \begin{bmatrix} \phi_r \\ \phi_i \end{bmatrix} \tag{A.16c}$$

$$\frac{\gamma}{\omega} \begin{bmatrix} \nabla \cdot \tilde{\mathbf{u}}_r & \nabla \cdot \tilde{\mathbf{u}}_i \end{bmatrix} \begin{bmatrix} 0 & 2 \\ -2 & 0 \end{bmatrix} \begin{bmatrix} \phi_r \\ \phi_i \end{bmatrix} \tag{A.16d}$$

$$(\nabla \cdot \tilde{\mathbf{u}}, f) + (f, \nabla \cdot \tilde{\mathbf{u}})$$

$$\begin{bmatrix} \nabla \cdot \tilde{\mathbf{u}}_r & \nabla \cdot \tilde{\mathbf{u}}_i \end{bmatrix} \begin{bmatrix} 1 & -i \\ i & 1 \end{bmatrix} \begin{bmatrix} f_r \\ f_i \end{bmatrix} + \begin{bmatrix} f_r & f_i \end{bmatrix} \begin{bmatrix} 1 & -i \\ i & 1 \end{bmatrix} \begin{bmatrix} \nabla \cdot \tilde{\mathbf{u}}_r \\ \nabla \cdot \tilde{\mathbf{u}}_i \end{bmatrix} \tag{A.17a}$$

$$\begin{bmatrix} \nabla \cdot \tilde{\mathbf{u}}_r & \nabla \cdot \tilde{\mathbf{u}}_i \end{bmatrix} \begin{bmatrix} 1 & -i \\ i & 1 \end{bmatrix} \begin{bmatrix} f_r \\ f_i \end{bmatrix} + \begin{bmatrix} \nabla \cdot \tilde{\mathbf{u}}_r & \nabla \cdot \tilde{\mathbf{u}}_i \end{bmatrix} \begin{bmatrix} 1 & i \\ -i & 1 \end{bmatrix} \begin{bmatrix} f_r \\ f_i \end{bmatrix} \tag{A.17b}$$

$$\begin{bmatrix} \nabla \cdot \tilde{\mathbf{u}}_r & \nabla \cdot \tilde{\mathbf{u}}_i \end{bmatrix} \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix} \begin{bmatrix} f_r \\ f_i \end{bmatrix} \tag{A.17c}$$

Continuous System

$$\omega\left(\tilde{\mathbf{u}}_r, \nabla\phi_i\right) + \omega^2\left(\tilde{\mathbf{u}}_r, \mathbf{u}_r\right) + \left(\nabla\cdot\tilde{\mathbf{u}}_r, \nabla\cdot\mathbf{u}_r\right) + \frac{\gamma}{\omega}\left(\nabla\cdot\tilde{\mathbf{u}}_r, \phi_i\right) = \left(\nabla\cdot\tilde{\mathbf{u}}_r, f_r\right) \tag{A.18a}$$

$$-\omega\left(\tilde{\mathbf{u}}_i, \nabla\phi_r\right) + \omega^2\left(\tilde{\mathbf{u}}_i, \mathbf{u}_i\right) + \left(\nabla\cdot\tilde{\mathbf{u}}_i, \nabla\cdot\mathbf{u}_i\right) - \frac{\gamma}{\omega}\left(\nabla\cdot\tilde{\mathbf{u}}_i, \phi_r\right) = \left(\nabla\cdot\tilde{\mathbf{u}}_i, f_i\right) \tag{A.18b}$$

Combined System

Continuous

$$\left(\frac{\gamma}{\omega}\right)^2\left(\tilde{\phi}_r, \phi_r\right) + \left(\nabla\tilde{\phi}_r, \nabla\phi_r\right) - \frac{\gamma}{\omega}\left(\tilde{\phi}_r, \nabla\cdot\mathbf{u}_i\right) - \omega\left(\nabla\tilde{\phi}_r, \mathbf{u}_i\right) = -\frac{\gamma}{\omega}\left(\tilde{\phi}_r, f_i\right) \tag{A.19a}$$

$$\left(\frac{\gamma}{\omega}\right)^2\left(\tilde{\phi}_i, \phi_i\right) + \left(\nabla\tilde{\phi}_i, \nabla\phi_i\right) + \frac{\gamma}{\omega}\left(\tilde{\phi}_i, \nabla\cdot\mathbf{u}_r\right) + \omega\left(\nabla\tilde{\phi}_i, \mathbf{u}_r\right) = \frac{\gamma}{\omega}\left(\tilde{\phi}_i, f_r\right) \tag{A.19b}$$

$$\omega\left(\tilde{\mathbf{u}}_r, \nabla\phi_i\right) + \omega^2\left(\tilde{\mathbf{u}}_r, \mathbf{u}_r\right) + \left(\nabla\cdot\tilde{\mathbf{u}}_r, \nabla\cdot\mathbf{u}_r\right) + \frac{\gamma}{\omega}\left(\nabla\cdot\tilde{\mathbf{u}}_r, \phi_i\right) = \left(\nabla\cdot\tilde{\mathbf{u}}_r, f_r\right) \tag{A.19c}$$

$$-\omega\left(\tilde{\mathbf{u}}_i, \nabla\phi_r\right) + \omega^2\left(\tilde{\mathbf{u}}_i, \mathbf{u}_i\right) + \left(\nabla\cdot\tilde{\mathbf{u}}_i, \nabla\cdot\mathbf{u}_i\right) - \frac{\gamma}{\omega}\left(\nabla\cdot\tilde{\mathbf{u}}_i, \phi_r\right) = \left(\nabla\cdot\tilde{\mathbf{u}}_i, f_i\right) \tag{A.19d}$$

$$\left(\frac{\gamma}{\omega}\right)^2 \mathfrak{Re}\left\{\left(\tilde{\phi}, \phi\right) + \left(\nabla\tilde{\phi}, \nabla\phi\right)\right\} + \mathfrak{Im}\left\{\frac{\gamma}{\omega}\left(\tilde{\phi}, \nabla\cdot\mathbf{u}\right) + \omega\left(\nabla\tilde{\phi}, \mathbf{u}\right)\right\} = \mathfrak{Im}\left\{\frac{\gamma}{\omega}\left(\tilde{\phi}, f\right)\right\} \tag{A.20a}$$

$$-\mathfrak{Im}\left\{\omega\left(\tilde{\mathbf{u}}, \nabla\phi\right) + \frac{\gamma}{\omega}\left(\nabla\cdot\tilde{\mathbf{u}}, \phi\right)\right\} + \mathfrak{Re}\left\{\omega^2\left(\tilde{\mathbf{u}}, \mathbf{u}\right) + \left(\nabla\cdot\tilde{\mathbf{u}}, \nabla\cdot\mathbf{u}\right)\right\} = \mathfrak{Re}\left\{\left(\nabla\cdot\tilde{\mathbf{u}}, f\right)\right\} \tag{A.20b}$$

$$(a, \mathrm{d}b) = \langle a, b\rangle - (\mathrm{d}a, b)$$

$$\left(\frac{\gamma}{\omega}\right)^2\left(\tilde{\phi}_r, \phi_r\right) + \left(\nabla\tilde{\phi}_r, \nabla\phi_r\right) - \left(\frac{\gamma}{\omega} - \omega\right)\left(\tilde{\phi}_r, \nabla\cdot\mathbf{u}_i\right) - \omega\left\langle\tilde{\phi}_r, \mathbf{u}_i\cdot\mathbf{n}\right\rangle = -\frac{\gamma}{\omega}\left(\tilde{\phi}_r, f_i\right) \tag{A.21a}$$

$$\left(\frac{\gamma}{\omega}\right)^2\left(\tilde{\phi}_i, \phi_i\right) + \left(\nabla\tilde{\phi}_i, \nabla\phi_i\right) + \left(\frac{\gamma}{\omega} - \omega\right)\left(\tilde{\phi}_r, \nabla\cdot\mathbf{u}_i\right) + \omega\left\langle\tilde{\phi}_r, \mathbf{u}_i\cdot\mathbf{n}\right\rangle = \frac{\gamma}{\omega}\left(\tilde{\phi}_i, f_r\right) \tag{A.21b}$$

$$\omega\left\langle\tilde{\mathbf{u}}_r\cdot\mathbf{n}, \phi_i\right\rangle + \omega^2\left(\tilde{\mathbf{u}}_r, \mathbf{u}_r\right) + \left(\nabla\cdot\tilde{\mathbf{u}}_r, \nabla\cdot\mathbf{u}_r\right) + \left(\frac{\gamma}{\omega} - \omega\right)\left(\nabla\cdot\tilde{\mathbf{u}}_r, \phi_i\right) = \left(\nabla\cdot\tilde{\mathbf{u}}_r, f_r\right) \tag{A.21c}$$

$$-\omega\left\langle\tilde{\mathbf{u}}_r\cdot\mathbf{n}, \phi_i\right\rangle + \omega^2\left(\tilde{\mathbf{u}}_i, \mathbf{u}_i\right) + \left(\nabla\cdot\tilde{\mathbf{u}}_i, \nabla\cdot\mathbf{u}_i\right) - \left(\frac{\gamma}{\omega} - \omega\right)\left(\nabla\cdot\tilde{\mathbf{u}}_i, \phi_r\right) = \left(\nabla\cdot\tilde{\mathbf{u}}_i, f_i\right) \tag{A.21d}$$

$$\mathfrak{Re}\left\{\left(\frac{\gamma}{\omega}\right)^2\left(\tilde{\phi},\phi\right)+\left(\nabla\tilde{\phi},\nabla\phi\right)\right\}+ \tag{A.22a}$$
$$\mathfrak{Im}\left\{\frac{\gamma}{\omega}\left(\tilde{\phi},\nabla\cdot\mathbf{u}\right)+\omega\left\langle\tilde{\phi},\mathbf{u}\cdot\mathbf{n}\right\rangle-\omega\left(\tilde{\phi},\nabla\cdot\mathbf{u}\right)\right\}=\mathfrak{Im}\left\{\frac{\gamma}{\omega}\left(\tilde{\phi},f\right)\right\}$$
$$-\mathfrak{Im}\left\{\omega\left\langle\tilde{\mathbf{u}}\cdot\mathbf{n},\phi\right\rangle-\omega\left(\nabla\cdot\tilde{\mathbf{u}},\phi\right)+\frac{\gamma}{\omega}\left(\nabla\cdot\tilde{\mathbf{u}},\phi\right)\right\}+ \tag{A.22b}$$
$$\mathfrak{Re}\left\{\omega^2\left(\tilde{\mathbf{u}},\mathbf{u}\right)+\left(\nabla\cdot\tilde{\mathbf{u}},\nabla\cdot\mathbf{u}\right)\right\}=\mathfrak{Re}\left\{\left(\nabla\cdot\tilde{\mathbf{u}},f\right)\right\}$$

$\mathbf{u}\cdot\mathbf{n}=i\omega\phi$ or $\phi=-\frac{i}{\omega}\mathbf{u}\cdot\mathbf{n}$

$$\mathfrak{Re}\left\{\omega^2\left[\left(\tilde{\phi},\phi\right)-\left\langle\tilde{\phi},\phi\right\rangle\right]+\left(\nabla\tilde{\phi},\nabla\phi\right)\right\}+ \tag{A.23a}$$
$$\mathfrak{Im}\left\{\frac{\gamma}{\omega}\left(\tilde{\phi},\nabla\cdot\mathbf{u}\right)-\omega\left(\tilde{\phi},\nabla\cdot\mathbf{u}\right)\right\}=\mathfrak{Im}\left\{\frac{\gamma}{\omega}\left(\tilde{\phi},f\right)\right\}$$
$$\mathfrak{Im}\left\{\omega\left(\nabla\cdot\tilde{\mathbf{u}},\phi\right)-\frac{\gamma}{\omega}\left(\nabla\cdot\tilde{\mathbf{u}},\phi\right)\right\}+ \tag{A.23b}$$
$$\mathfrak{Re}\left\{\omega^2\left(\tilde{\mathbf{u}},\mathbf{u}\right)+\left(\nabla\cdot\tilde{\mathbf{u}},\nabla\cdot\mathbf{u}\right)-\left\langle\tilde{\mathbf{u}}\cdot\mathbf{n},\mathbf{u}\cdot\mathbf{n}\right\rangle\right\}=\mathfrak{Re}\left\{\left(\nabla\cdot\tilde{\mathbf{u}},f\right)\right\}$$

Discrete

$$\tilde{\phi}_r:\left[\left(\frac{\gamma}{\omega}\right)^2\mathbb{M}_{00}+\mathbb{E}_{10}^T\mathbb{M}_{11}\mathbb{E}_{10}\right]\left[\phi_r\right]-\left[\frac{\gamma}{\omega}\mathbb{M}_{0\bar{2}}\mathbb{E}_{\bar{2}\bar{1}}+\omega\mathbb{E}_{10}^T\mathbb{M}_{1\bar{1}}\right]\left[\mathbf{u}_i\right]=-\frac{\gamma}{\omega}\mathbb{M}_{00}\left[f_i\right] \tag{A.24a}$$
$$\tilde{\phi}_i:\left[\left(\frac{\gamma}{\omega}\right)^2\mathbb{M}_{00}+\mathbb{E}_{10}^T\mathbb{M}_{11}\mathbb{E}_{10}\right]\left[\phi_i\right]+\left[\frac{\gamma}{\omega}\mathbb{M}_{0\bar{2}}\mathbb{E}_{\bar{2}\bar{1}}+\omega\mathbb{E}_{10}^T\mathbb{M}_{1\bar{1}}\right]\left[\mathbf{u}_r\right]=\frac{\gamma}{\omega}\mathbb{M}_{00}\left[f_r\right] \tag{A.24b}$$
$$\tilde{\mathbf{u}}_r:\left[\omega\mathbb{M}_{\bar{1}1}\mathbb{E}_{10}+\frac{\gamma}{\omega}\mathbb{E}_{\bar{2}\bar{1}}^T\mathbb{M}_{\bar{2}0}\right]\left[\phi_i\right]+\left[\omega^2\mathbb{M}_{\bar{1}\bar{1}}+\mathbb{E}_{\bar{2}\bar{1}}^T\mathbb{M}_{\bar{2}\bar{2}}\mathbb{E}_{\bar{2}\bar{1}}\right]\left[\mathbf{u}_r\right]=\mathbb{E}_{\bar{2}\bar{1}}^T\mathbb{M}_{\bar{2}0}\left[f_r\right] \tag{A.24c}$$
$$\tilde{\mathbf{u}}_i:-\left[\omega\mathbb{M}_{\bar{1}1}\mathbb{E}_{10}+\frac{\gamma}{\omega}\mathbb{E}_{\bar{2}\bar{1}}^T\mathbb{M}_{\bar{2}0}\right]\left[\phi_r\right]+\left[\omega^2\mathbb{M}_{\bar{1}\bar{1}}+\mathbb{E}_{\bar{2}\bar{1}}^T\mathbb{M}_{\bar{2}\bar{2}}\mathbb{E}_{\bar{2}\bar{1}}\right]\left[\mathbf{u}_i\right]=\mathbb{E}_{\bar{2}\bar{1}}^T\mathbb{M}_{\bar{2}0}\left[f_i\right] \tag{A.24d}$$

$(a, \mathrm{d}b) = \langle a, b\rangle - (\mathrm{d}a, b)$

$$\tilde{\phi}_r : \left[\left(\frac{\gamma}{\omega}\right)^2 \mathbb{M}_{00} + \mathbb{E}_{10}^T \mathbb{M}_{11} \mathbb{E}_{10}\right][\phi_r] - \left[\left(\frac{\gamma}{\omega} - \omega\right)\mathbb{M}_{0\bar{2}}\mathbb{E}_{\bar{2}\bar{1}} + \omega \mathbb{M}_{0\bar{1}}^{BC}\right][\mathbf{u}_i] = -\frac{\gamma}{\omega}\mathbb{M}_{00}[f_i] \tag{A.25a}$$

$$\tilde{\phi}_i : \left[\left(\frac{\gamma}{\omega}\right)^2 \mathbb{M}_{00} + \mathbb{E}_{10}^T \mathbb{M}_{11} \mathbb{E}_{10}\right][\phi_i] + \left[\left(\frac{\gamma}{\omega} - \omega\right)\mathbb{M}_{0\bar{2}}\mathbb{E}_{\bar{2}\bar{1}} + \omega \mathbb{M}_{0\bar{1}}^{BC}\right][\mathbf{u}_r] = \frac{\gamma}{\omega}\mathbb{M}_{00}[f_r] \tag{A.25b}$$

$$\tilde{\mathbf{u}}_r : \left[\omega \mathbb{M}_{\bar{1}0}^{BC} + \left(\frac{\gamma}{\omega} - \omega\right)\mathbb{E}_{\bar{2}\bar{1}}^T \mathbb{M}_{\bar{2}0}\right][\phi_i] + \left[\omega^2 \mathbb{M}_{\bar{1}\bar{1}} + \mathbb{E}_{\bar{2}\bar{1}}^T \mathbb{M}_{\bar{2}\bar{2}}\mathbb{E}_{\bar{2}\bar{1}}\right][\mathbf{u}_r] = \mathbb{E}_{\bar{2}\bar{1}}^T \mathbb{M}_{\bar{2}0}[f_r] \tag{A.25c}$$

$$\tilde{\mathbf{u}}_i : -\left[\omega \mathbb{M}_{\bar{1}0}^{BC} + \left(\frac{\gamma}{\omega} - \omega\right)\mathbb{E}_{\bar{2}\bar{1}}^T \mathbb{M}_{\bar{2}0}\right][\phi_r] + \left[\omega^2 \mathbb{M}_{\bar{1}\bar{1}} + \mathbb{E}_{\bar{2}\bar{1}}^T \mathbb{M}_{\bar{2}\bar{2}}\mathbb{E}_{\bar{2}\bar{1}}\right][\mathbf{u}_i] = \mathbb{E}_{\bar{2}\bar{1}}^T \mathbb{M}_{\bar{2}0}[f_i] \tag{A.25d}$$

Boundary Condition

$$\lim_{r\to\infty} |r|^{\frac{d-1}{2}} \left(\frac{\partial \phi}{\partial r} - i\omega\phi\right) = -i\omega g \tag{A.26a}$$

$$\frac{\partial \phi}{\partial r} - i\omega\phi = -i\omega g \tag{A.26b}$$

$$\nabla_r \phi - i\omega\phi = -i\omega g \tag{A.26c}$$

$$\frac{\partial x}{\partial r}\nabla_x \phi + \frac{\partial y}{\partial r}\nabla_y \phi - i\omega\phi = -i\omega g \tag{A.26d}$$

$$\cos\theta \nabla_x \phi + \sin\theta \nabla_y \phi - i\omega\phi = -i\omega g \tag{A.26e}$$

(A.26f)

Problem specific discretizations

Demkowicz Problem A

$\frac{\partial \phi}{\partial n} - i\omega\phi = -i\omega g$ or $\mathbf{u}\cdot\mathbf{n} + \phi = g$ on all boundaries.

$$\mathfrak{Re}\left\{\left(\frac{\gamma}{\omega}\right)^2 \left(\tilde{\phi}, \phi\right) + \left(\nabla\tilde{\phi}, \nabla\phi\right)\right\} + \tag{A.27a}$$

$$\mathfrak{Im}\left\{\frac{\gamma}{\omega}\left(\tilde{\phi}, \nabla\cdot\mathbf{u}\right) + \omega\left\langle\tilde{\phi}, \mathbf{u}\cdot\mathbf{n}\right\rangle - \omega\left(\tilde{\phi}, \nabla\cdot\mathbf{u}\right)\right\} = \mathfrak{Im}\left\{\frac{\gamma}{\omega}\left(\tilde{\phi}, f\right)\right\}$$

$$-\mathfrak{Im}\left\{\omega\left\langle\tilde{\mathbf{u}}\cdot\mathbf{n}, \phi\right\rangle - \omega\left(\nabla\cdot\tilde{\mathbf{u}}, \phi\right) + \frac{\gamma}{\omega}\left(\nabla\cdot\tilde{\mathbf{u}}, \phi\right)\right\} + \tag{A.27b}$$

$$\mathfrak{Re}\left\{\omega^2\left(\tilde{\mathbf{u}}, \mathbf{u}\right) + \left(\nabla\cdot\tilde{\mathbf{u}}, \nabla\cdot\mathbf{u}\right)\right\} = \mathfrak{Re}\left\{\left(\nabla\cdot\tilde{\mathbf{u}}, f\right)\right\}$$

$$\begin{aligned} &\mathfrak{Re}\left\{\omega^2\left(\tilde{\phi},\phi\right)+\left(\nabla\tilde{\phi},\nabla\phi\right)\right\}+ \\ \mathfrak{Im}\left\{\frac{\gamma}{\omega}\left(\tilde{\phi},\nabla\cdot\mathbf{u}\right)-\omega\left\langle\tilde{\phi},\phi\right\rangle-\omega\left(\tilde{\phi},\nabla\cdot\mathbf{u}\right)\right\} &= \mathfrak{Im}\left\{\frac{\gamma}{\omega}\left(\tilde{\phi},f\right)-\omega\left\langle\tilde{\phi},g\right\rangle\right\} \end{aligned} \tag{A.28a}$$

$$\begin{aligned} &\mathfrak{Im}\left\{\omega\left\langle\tilde{\mathbf{u}}\cdot\mathbf{n},\mathbf{u}\cdot\mathbf{n}\right\rangle+\omega\left(\nabla\cdot\tilde{\mathbf{u}},\phi\right)-\frac{\gamma}{\omega}\left(\nabla\cdot\tilde{\mathbf{u}},\phi\right)\right\}+ \\ \mathfrak{Re}\left\{\omega^2\left(\tilde{\mathbf{u}},\mathbf{u}\right)+\left(\nabla\cdot\tilde{\mathbf{u}},\nabla\cdot\mathbf{u}\right)\right\} &= \mathfrak{Re}\left\{\left(\nabla\cdot\tilde{\mathbf{u}},f\right)\right\}+\mathfrak{Im}\left\{\omega\left\langle\tilde{\mathbf{u}}\cdot\mathbf{n},g\right\rangle\right\} \end{aligned} \tag{A.28b}$$

Demkowicz Problem A (Directional Derivative)

$\nabla_r\phi - i\omega\phi = -i\omega g$ or $\nabla_n\phi\cdot\frac{\mathbf{r}}{\mathbf{n}}+\nabla_t\phi\cdot\frac{\mathbf{r}}{\mathbf{t}}-i\omega\phi=-i\omega g$ or $\mathbf{u}\cdot\mathbf{n}\cdot\frac{\mathbf{r}}{\mathbf{n}}+\phi=g$ on left and bottom boundary.
$\nabla\phi\cdot\mathbf{n}\cdot\frac{\mathbf{r}}{\mathbf{n}}-i\omega\phi=0$ or $\mathbf{u}\cdot\mathbf{n}\cdot\frac{\mathbf{r}}{\mathbf{n}}+\phi=0$ on right and top boundary.

$$\nabla_r\phi=\nabla\phi\cdot\mathbf{r}=\nabla_x\phi\cdot\mathbf{r}_x+\nabla_y\phi\cdot\mathbf{r}_y= \tag{A.29a}$$
$$(\nabla_n\phi\cdot\mathbf{n}_x+\nabla_t\phi\cdot\mathbf{t}_x)\cdot\mathbf{r}_x+(\nabla_n\phi\cdot\mathbf{n}_y+\nabla_t\phi\cdot\mathbf{t}_y)\cdot\mathbf{r}_y= \tag{A.29b}$$
$$\nabla_n\phi\,(\mathbf{n}\cdot\mathbf{r})+\nabla_t\phi\,(\mathbf{t}\cdot\mathbf{r}) \tag{A.29c}$$

$$\nabla_r\phi-i\omega\phi=-i\omega g \tag{A.30a}$$
$$\nabla_n\phi\,(\mathbf{n}\cdot\mathbf{r})+\nabla_t\phi\,(\mathbf{t}\cdot\mathbf{r})-i\omega\phi=-i\omega g \tag{A.30b}$$
$$\mathbf{u}\cdot\mathbf{n}\,(\mathbf{n}\cdot\mathbf{r})+\mathbf{v}\cdot\mathbf{t}\frac{\mathbf{t}\cdot\mathbf{r}}{i\omega}+\phi=g \tag{A.30c}$$

Left: $\mathbf{n}=\begin{pmatrix}-1\\0\end{pmatrix}$, $\mathbf{t}=\begin{pmatrix}0\\1\end{pmatrix}$, $\mathbf{r}=\begin{pmatrix}\cos\theta\\\sin\theta\end{pmatrix}$ and $g=g$

$$-\mathbf{u}\cdot\mathbf{n}\cos\theta+\mathbf{v}\cdot\mathbf{t}\frac{\sin\theta}{i\omega}+\phi=g,\text{ where }\mathbf{v}\cdot\mathbf{t}=+\mathbf{v}_y=-\mathbb{E}_{10,y}\phi \tag{A.31}$$

Bottom: $\mathbf{n}=\begin{pmatrix}0\\-1\end{pmatrix}$, $\mathbf{t}=\begin{pmatrix}1\\0\end{pmatrix}$, $\mathbf{r}=\begin{pmatrix}\cos\theta\\\sin\theta\end{pmatrix}$ and $g=g$

$$-\mathbf{u}\cdot\mathbf{n}\sin\theta+\mathbf{v}\cdot\mathbf{t}\frac{\cos\theta}{i\omega}+\phi=g,\text{ where }\mathbf{v}\cdot\mathbf{t}=+\mathbf{v}_x=-\mathbb{E}_{10,x}\phi \tag{A.32}$$

Right: $\mathbf{n}=\begin{pmatrix}1\\0\end{pmatrix}$, $\mathbf{t}=\begin{pmatrix}0\\1\end{pmatrix}$, $\mathbf{r}=\begin{pmatrix}\cos\theta\\\sin\theta\end{pmatrix}$ and $g=0$

$$\mathbf{u}\cdot\mathbf{n}\cos\theta+\mathbf{v}\cdot\mathbf{t}\frac{\sin\theta}{i\omega}+\phi=0,\text{ where }\mathbf{v}\cdot\mathbf{t}=+\mathbf{v}_y=-\mathbb{E}_{10,y}\phi \tag{A.33}$$

Top: $\mathbf{n} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$, $\mathbf{t} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$, $\mathbf{r} = \begin{pmatrix} \cos\theta \\ \sin\theta \end{pmatrix}$ and $g = 0$

$$\mathbf{u} \cdot \mathbf{n} \sin\theta + \mathbf{v} \cdot \mathbf{t} \frac{\cos\theta}{i\omega} + \phi = 0, \text{ where } \mathbf{v} \cdot \mathbf{t} = +\mathbf{v}_x = -\mathbb{E}_{10,x}\phi \tag{A.34}$$

$$\mathfrak{Re}\left\{\left(\frac{\gamma}{\omega}\right)^2\left(\tilde{\phi},\phi\right)+\left(\nabla\tilde{\phi},\nabla\phi\right)\right\}+ \tag{A.35a}$$
$$\mathfrak{Im}\left\{\frac{\gamma}{\omega}\left(\tilde{\phi},\nabla\cdot\mathbf{u}\right)+\omega\left\langle\tilde{\phi},\mathbf{u}\cdot\mathbf{n}\right\rangle-\omega\left(\tilde{\phi},\nabla\cdot\mathbf{u}\right)\right\}=\mathfrak{Im}\left\{\frac{\gamma}{\omega}\left(\tilde{\phi},f\right)\right\}$$
$$-\mathfrak{Im}\left\{\omega\left\langle\tilde{\mathbf{u}}\cdot\mathbf{n},\phi\right\rangle-\omega\left(\nabla\cdot\tilde{\mathbf{u}},\phi\right)+\frac{\gamma}{\omega}\left(\nabla\cdot\tilde{\mathbf{u}},\phi\right)\right\}+ \tag{A.35b}$$
$$\mathfrak{Re}\left\{\omega^2\left(\tilde{\mathbf{u}},\mathbf{u}\right)+\left(\nabla\cdot\tilde{\mathbf{u}},\nabla\cdot\mathbf{u}\right)\right\}=\mathfrak{Re}\left\{\left(\nabla\cdot\tilde{\mathbf{u}},f\right)\right\}$$

$$\mathfrak{Re}\left\{\omega^2\left(\tilde{\phi},\phi\right)+\left(\nabla\tilde{\phi},\nabla\phi\right)\right\}+$$
$$\mathfrak{Im}\left\{\frac{\gamma}{\omega}\left(\tilde{\phi},\nabla\cdot\mathbf{u}\right)-\omega\left\langle\tilde{\phi},\mathbf{v}\cdot\mathbf{t}\frac{\mathbf{t}\cdot\mathbf{r}}{i\omega\mathbf{n}\cdot\mathbf{r}}\right\rangle-\omega\left\langle\tilde{\phi},\frac{\phi}{\mathbf{n}\cdot\mathbf{r}}\right\rangle-\omega\left(\tilde{\phi},\nabla\cdot\mathbf{u}\right)\right\}= \tag{A.36a}$$
$$\mathfrak{Im}\left\{\frac{\gamma}{\omega}\left(\tilde{\phi},f\right)-\omega\left\langle\tilde{\phi},\frac{g}{\mathbf{n}\cdot\mathbf{r}}\right\rangle\right\}$$
$$\mathfrak{Im}\left\{\omega\left\langle\tilde{\mathbf{u}}\cdot\mathbf{n},\mathbf{u}\cdot\mathbf{n}\left(\mathbf{n}\cdot\mathbf{r}\right)\right\rangle+\omega\left\langle\tilde{\mathbf{u}}\cdot\mathbf{n},\mathbf{v}\cdot\mathbf{t}\frac{\mathbf{t}\cdot\mathbf{r}}{i\omega}\right\rangle+\omega\left(\nabla\cdot\tilde{\mathbf{u}},\phi\right)-\frac{\gamma}{\omega}\left(\nabla\cdot\tilde{\mathbf{u}},\phi\right)\right\}+$$
$$\mathfrak{Re}\left\{\omega^2\left(\tilde{\mathbf{u}},\mathbf{u}\right)+\left(\nabla\cdot\tilde{\mathbf{u}},\nabla\cdot\mathbf{u}\right)\right\}=$$
$$\mathfrak{Re}\left\{\left(\nabla\cdot\tilde{\mathbf{u}},f\right)\right\}+\mathfrak{Im}\left\{\omega\left\langle\tilde{\mathbf{u}}\cdot\mathbf{n},g\right\rangle\right\} \tag{A.36b}$$

$$\mathfrak{Re}\left\{\omega^2\left(\tilde{\phi},\phi\right)+\left(\nabla\tilde{\phi},\nabla\phi\right)-\left\langle\tilde{\phi},\mathbf{v}\cdot\mathbf{t}\frac{\mathbf{t}\cdot\mathbf{r}}{\mathbf{n}\cdot\mathbf{r}}\right\rangle\right\}+$$
$$\mathfrak{Im}\left\{\frac{\gamma}{\omega}\left(\tilde{\phi},\nabla\cdot\mathbf{u}\right)-\omega\left\langle\tilde{\phi},\frac{\phi}{\mathbf{n}\cdot\mathbf{r}}\right\rangle-\omega\left(\tilde{\phi},\nabla\cdot\mathbf{u}\right)\right\}= \tag{A.37a}$$
$$\mathfrak{Im}\left\{\frac{\gamma}{\omega}\left(\tilde{\phi},f\right)-\omega\left\langle\tilde{\phi},\frac{g}{\mathbf{n}\cdot\mathbf{r}}\right\rangle\right\}$$
$$\mathfrak{Im}\left\{\omega\left\langle\tilde{\mathbf{u}}\cdot\mathbf{n},\mathbf{u}\cdot\mathbf{n}\left(\mathbf{n}\cdot\mathbf{r}\right)\right\rangle+\omega\left(\nabla\cdot\tilde{\mathbf{u}},\phi\right)-\frac{\gamma}{\omega}\left(\nabla\cdot\tilde{\mathbf{u}},\phi\right)\right\}+$$
$$\mathfrak{Re}\left\{\left\langle\tilde{\mathbf{u}}\cdot\mathbf{n},\mathbf{v}\cdot\mathbf{t}\left(\mathbf{t}\cdot\mathbf{r}\right)\right\rangle+\omega^2\left(\tilde{\mathbf{u}},\mathbf{u}\right)+\left(\nabla\cdot\tilde{\mathbf{u}},\nabla\cdot\mathbf{u}\right)\right\}= \tag{A.37b}$$
$$\mathfrak{Re}\left\{\left(\nabla\cdot\tilde{\mathbf{u}},f\right)\right\}+\mathfrak{Im}\left\{\omega\left\langle\tilde{\mathbf{u}}\cdot\mathbf{n},g\right\rangle\right\}$$

Discrete System

$$\begin{bmatrix} \left[\left(\frac{\gamma}{\omega}\right)^2 \mathbb{M}_{00} + \mathbb{E}_{\bar{1}0}^T \mathbb{M}_{11} \mathbb{E}_{\bar{1}0}\right] & 0 & 0 & -\left[\frac{\gamma}{\omega}\mathbb{M}_{0\bar{2}}\mathbb{E}_{\bar{2}\bar{1}} + \omega \mathbb{E}_{\bar{1}0}^T \mathbb{M}_{1\bar{1}}\right] \\ 0 & \left[\left(\frac{\gamma}{\omega}\right)^2 \mathbb{M}_{00} + \mathbb{E}_{\bar{1}0}^T \mathbb{M}_{11} \mathbb{E}_{\bar{1}0}\right] & \left[\frac{\gamma}{\omega}\mathbb{M}_{0\bar{2}}\mathbb{E}_{\bar{2}\bar{1}} + \omega \mathbb{E}_{\bar{1}0}^T \mathbb{M}_{1\bar{1}}\right] & 0 \\ & \left[\omega \mathbb{M}_{\bar{1}1}\mathbb{E}_{10} + \frac{\gamma}{\omega}\mathbb{E}_{\bar{2}\bar{1}}^T \mathbb{M}_{\bar{2}0}\right] & \left[\omega^2 \mathbb{M}_{\bar{1}\bar{1}} + \mathbb{E}_{\bar{2}\bar{1}}^T \mathbb{M}_{\bar{2}\bar{2}}\mathbb{E}_{\bar{2}\bar{1}}\right] & 0 \\ -\left[\omega \mathbb{M}_{\bar{1}1}\mathbb{E}_{10} + \frac{\gamma}{\omega}\mathbb{E}_{\bar{2}\bar{1}}^T \mathbb{M}_{\bar{2}0}\right] & 0 & 0 & \left[\omega^2 \mathbb{M}_{\bar{1}\bar{1}} + \mathbb{E}_{\bar{2}\bar{1}}^T \mathbb{M}_{\bar{2}\bar{2}}\mathbb{E}_{\bar{2}\bar{1}}\right] \end{bmatrix} \begin{bmatrix} \phi_r \\ \phi_i \\ \mathbf{u}_r \\ \mathbf{u}_i \end{bmatrix} = \begin{bmatrix} -\frac{\gamma}{\omega}\mathbb{M}_{00}\left[f_i\right] \\ \frac{\gamma}{\omega}\mathbb{M}_{00}\left[f_r\right] \\ \mathbb{E}_{\bar{2}\bar{1}}^T \mathbb{M}_{\bar{2}0}\left[f_r\right] \\ \mathbb{E}_{\bar{2}\bar{1}}^T \mathbb{M}_{\bar{2}0}\left[f_i\right] \end{bmatrix} \tag{A.38a}$$

$$\begin{bmatrix} \left[\left(\frac{\gamma}{\omega}\right)^2 \mathbb{M}_{00} + \mathbb{E}_{\bar{1}0}^T \mathbb{M}_{11} \mathbb{E}_{\bar{1}0}\right] & 0 & 0 & -\left[\left(\frac{\gamma}{\omega} - \omega\right)\mathbb{M}_{0\bar{2}}\mathbb{E}_{\bar{2}\bar{1}} + \omega \mathbb{M}_{0\bar{1}}^{BC}\right] \\ 0 & \left[\left(\frac{\gamma}{\omega}\right)^2 \mathbb{M}_{00} + \mathbb{E}_{\bar{1}0}^T \mathbb{M}_{11} \mathbb{E}_{\bar{1}0}\right] & \left[\left(\frac{\gamma}{\omega} - \omega\right)\mathbb{M}_{0\bar{2}}\mathbb{E}_{\bar{2}\bar{1}} + \omega \mathbb{M}_{0\bar{1}}^{BC}\right] & 0 \\ & \left[\omega \mathbb{M}_{\bar{1}0}^{BC} + \left(\frac{\gamma}{\omega} - \omega\right)\mathbb{E}_{\bar{2}\bar{1}}^T \mathbb{M}_{\bar{2}0}\right] & \left[\omega^2 \mathbb{M}_{\bar{1}\bar{1}} + \mathbb{E}_{\bar{2}\bar{1}}^T \mathbb{M}_{\bar{2}\bar{2}}\mathbb{E}_{\bar{2}\bar{1}}\right] & 0 \\ -\left[\omega \mathbb{M}_{\bar{1}0}^{BC} + \left(\frac{\gamma}{\omega} - \omega\right)\mathbb{E}_{\bar{2}\bar{1}}^T \mathbb{M}_{\bar{2}0}\right] & 0 & 0 & \left[\omega^2 \mathbb{M}_{\bar{1}\bar{1}} + \mathbb{E}_{\bar{2}\bar{1}}^T \mathbb{M}_{\bar{2}\bar{2}}\mathbb{E}_{\bar{2}\bar{1}}\right] \end{bmatrix} \begin{bmatrix} \phi_r \\ \phi_i \\ \mathbf{u}_r \\ \mathbf{u}_i \end{bmatrix} = \begin{bmatrix} -\frac{\gamma}{\omega}\mathbb{M}_{00}\left[f_i\right] \\ \frac{\gamma}{\omega}\mathbb{M}_{00}\left[f_r\right] \\ \mathbb{E}_{\bar{2}\bar{1}}^T \mathbb{M}_{\bar{2}0}\left[f_r\right] \\ \mathbb{E}_{\bar{2}\bar{1}}^T \mathbb{M}_{\bar{2}0}\left[f_i\right] \end{bmatrix} \tag{A.39a}$$

Appendix B

Matlab Code

B.1 Diffraction.m

```
%% Misc
% other.M00 = M00; other.M02d = M02d; other.M11 = M11; other.M11d = M11d;
    other.M1d1d = M1d1d; other.M2d2d = M2d2d; other.M00_BC = M00_BC; other.M01_BC
    = M01_BC; other.M11_BC = M11_BC; other.E10 = E10; other.E2d1d = E2d1d;
    other.E1d0d = E1d0d; other.E21 = E21;
% other.err = err;
clc;
clearvars -except other ConvCheck
close all

addpath('../functions/Hermite')
addpath('../functions')

%% Define Parameters
count=0;
% Methodological
for nDeriv = 0
P = 4*(1-nDeriv);
gam = -1; % HelmHoltz or Reaction-Diffusion
eqn = 0; % 0:1-eqn fem, 1:2-eqn fem, 2:2-eqn lsfem
A = 2; % A: 1:interference, 2:periodic left, 3:No periodic, 4:prescribe U left
for Nwv = 4 %2.^(1:7); % Number of elements per wave
wvnr = 8;
    clearvars -except ConvCheck P nDeriv wvnr Nwv ERR count other eqn gam A
    disp([nDeriv P Nwv wvnr])

% Practical
diffr = 1; % Diffraction model or wave propagation model
plane_wave = 0; % Plane wave model else real sin()sin() model
integr_U = floor(A/4); % Interpolate U for specification normal velocity
L = 1.5; % Distance from opening to analyze result
theta = 0*pi/4;
```

```
plotje = 1;
anim = 1;
saveFig = 0;
domain.x = [0 1+1*diffr];
domain.y = [0 1];
xf = 0.25;   % relative position slit
yf = 1*diffr/wvnr/diff(domain.y);   % relative size slit
plot_bl = 1:1:6; % Which blocks to plot
% num_i = 2*(P+1)*(nDeriv+1); % number of grid point on fine mesh
a = ceil(200/(wvnr*Nwv*diff(domain.x)*P+1));
num_i = 10;%a*P+1; clear a;

k = wvnr*2*pi; % omega=2*pi is one wave on domain [0,1]
domain.dx = diff(domain.x);
domain.dy = diff(domain.y);
domain.xs = [domain.x(1);domain.x(1)+domain.dx*xf;domain.x(2)];
domain.ys = [domain.y(1);domain.y(1)+domain.dy*(1-yf)/2;domain.y(2)-domain.dy*(1-
    yf)/2;domain.y(2)];
N = round(Nwv*wvnr*domain.dx); % omega*h=pi/2 4 elements per wavelength
H = domain.dx/N; % mesh size
M = round(domain.dy/H);
MESH.domain = domain; MESH.N = N; MESH.M = M; MESH.P = P; MESH.nDeriv = nDeriv;

%% Loop mesh blocks
mesh = cell(6,1); M00 = mesh; M11 = mesh; M2d2d = mesh; M02d = mesh; M11d = mesh;
     M1d1d = mesh;  M11x = mesh; M11y = mesh; bc = mesh; n2 = mesh; n1 = mesh;
    M1d1dx = mesh; M1d1dy = mesh; M11dxy = mesh; M11dyx = mesh;
MESH.X = []; MESH.Y = []; MESH.XY = []; MESH.XYt = []; MESH.edgesx = [];
    MESH.edgesy = []; MESH.volumes = [];
for block_y = 1:3
    for block_x = 1:2
        %% Calculate block general data
        block = block_x+2*(block_y-1);
        domain.x = [domain.xs(block_x) domain.xs(block_x+1)]; domain.X(block,:) =
             domain.x;
        domain.y = [domain.ys(block_y) domain.ys(block_y+1)]; domain.Y(block,:) =
             domain.y;
        dx(block) = diff(domain.X(block,:));
        dy(block) = diff(domain.Y(block,:));
        N = max(round(dx(block)/H),1);
        M = max(round(dy(block)/H),1);
        h=H/P;
        mesh{block} = sizes(N,M,P);
        mesh{block}.domain = domain;
        mesh{block}.nDeriv = nDeriv;
        nn_g = mesh{block}.nn_g; ne_g_h = mesh{block}.ne_g_h; ne_g_v = mesh{block
            }.ne_g_v; ne_g = mesh{block}.ne_g; nv_g = mesh{block}.nv_g;

        %% Create Mass Matrices
        [M00{block},M11{block},M2d2d{block},M02d{block},M11d{block},M1d1d{block}]
             = mass_matrices2D_v4_2(P,N,M,nDeriv,domain); %
        M11x{block} = M11{block}(1:(ne_g_h+nDeriv*(ne_g_h+nn_g*2)),1:(ne_g_h+
            nDeriv*(ne_g_h+nn_g*2)));
        M11y{block} = M11{block}((ne_g_h+nDeriv*(ne_g_h+nn_g*2))+1:end,(ne_g_h+
            nDeriv*(ne_g_h+nn_g*2))+1:end);
        M11dxy{block} = M11d{block}(1:(ne_g_h+nDeriv*(ne_g_h+nn_g*2)),1:(ne_g_v+
            nDeriv*(ne_g_v+nn_g*2)));
```

```
M11dyx{block} = M11d{block}((ne_g_h+nDeriv*(ne_g_h+nn_g*2))+1:end,(ne_g_v
    +nDeriv*(ne_g_v+nn_g*2))+1:end);
M1d1dy{block} = M1d1d{block}(1:(ne_g_v+nDeriv*(ne_g_v+nn_g*2)),1:(ne_g_v+
    nDeriv*(ne_g_v+nn_g*2)));
M1d1dx{block} = M1d1d{block}((ne_g_v+nDeriv*(ne_g_v+nn_g*2))+1:end,(
    ne_g_v+nDeriv*(ne_g_v+nn_g*2))+1:end);

%% Element spacing
% Main elements
mesh{block}.xN = linspace(domain.x(1),domain.x(2),N+1);
mesh{block}.yN = linspace(domain.y(1),domain.y(2),M+1);
% Normal and fine local GLL nodes
mesh{block}.xGLL = ((GLLnodes(P)+1)/2)/N*(domain.x(2)-domain.x(1)); mesh{
    block}.XGLL = 0;
mesh{block}.xfGLL = ((GLLnodes(num_i-1)+1)/2)/N*(domain.x(2)-domain.x(1))
    ; mesh{block}.XfGLL = 0;
% Normal and fine global GLL nodes
for iN=1:N
    mesh{block}.XGLL = [mesh{block}.XGLL(1:end-1) mesh{block}.xGLL+mesh{
        block}.xN(iN)];
    mesh{block}.XfGLL = [mesh{block}.XfGLL(1:end-1) mesh{block}.xfGLL+
        mesh{block}.xN(iN)];
end
mesh{block}.yGLL = ((GLLnodes(P)+1)/2)/M*(domain.y(2)-domain.y(1)); mesh{
    block}.YGLL = 0;
mesh{block}.yfGLL = ((GLLnodes(num_i-1)+1)/2)/M*(domain.y(2)-domain.y(1))
    ; mesh{block}.YfGLL = 0;
for jM=1:M
    mesh{block}.YGLL = [mesh{block}.YGLL(1:end-1) mesh{block}.yGLL+mesh{
        block}.yN(jM)];
    mesh{block}.YfGLL = [mesh{block}.YfGLL(1:end-1) mesh{block}.yfGLL+
        mesh{block}.yN(jM)];
end
% Grid nodes
[mesh{block}.XGrid, mesh{block}.YGrid] = meshgrid(mesh{block}.XGLL,mesh{
    block}.YGLL);
[mesh{block}.XfGrid, mesh{block}.YfGrid] = meshgrid(mesh{block}.XfGLL,
    mesh{block}.YfGLL);
mesh{block}.X = mesh{block}.XGrid'; mesh{block}.X = mesh{block}.X(:);
mesh{block}.Y = mesh{block}.YGrid'; mesh{block}.Y = mesh{block}.Y(:);
mesh{block}.Xf = mesh{block}.XfGrid'; mesh{block}.Xf = mesh{block}.Xf(:);
mesh{block}.Yf = mesh{block}.YfGrid'; mesh{block}.Yf = mesh{block}.Yf(:);
mesh{block}.XY = [mesh{block}.X mesh{block}.Y];

%% Assemble mesh data
nodes = (1:nn_g)'; node1 = nodes; node2 = nodes;
node1((1:M*P+1)*(N*P+1)) = [];
node2((0:M*P)*(N*P+1)+1) = [];
mesh{block}.edgesx = [node1 node2 (mesh{block}.X(node1)+mesh{block}.X(
    node2))/2 mesh{block}.Y(node1)];
node3 = (1:(nn_g-(N*P+1)))'; node4 = ((N*P+2):nn_g)';
mesh{block}.edgesy = [node3 node4 mesh{block}.X(node3) (mesh{block}.Y(
    node3)+mesh{block}.Y(node4))/2];
clearvars -regexp node[1234s]
edgesy = (1:ne_g_v)'; edge3 = edgesy; edge4 = edgesy;
edge1 = (1:(ne_g_h-N*P))'; edge2 = ((N*P+1):ne_g_h)';
edge3((1:M*P)*(N*P+1)) = [];
```

```
        edge4((0:(M*P-1))*(N*P+1)+1) = [];
        mesh{block}.volumes = [edge1 edge2 edge3 edge4 mesh{block}.edgesx(edge1
            ,3) mesh{block}.edgesy(edge3,4)];
        clearvars -regexp edge[1234{sy}]
        MESH.XY = [MESH.XY; mesh{block}.XY];
        MESH.X = [MESH.X; mesh{block}.X];
        MESH.Y = [MESH.Y; mesh{block}.Y];
        if block>1
            mesh{block}.nn_g_cs = mesh{block-1}.nn_g_cs+nn_g;
            mesh{block}.ne_g_cs = mesh{block-1}.ne_g_cs+ne_g;
            mesh{block}.ne_g_h_cs = mesh{block-1}.ne_g_h_cs+ne_g_h;
            mesh{block}.ne_g_v_cs = mesh{block-1}.ne_g_v_cs+ne_g_v;
            mesh{block}.nv_g_cs = mesh{block-1}.nv_g_cs+nv_g;
            MESH.edgesx = [MESH.edgesx; [mesh{block}.edgesx(:,1:2)+mesh{block-1}
                .nn_g_cs mesh{block}.edgesx(:,3:4)]];
            MESH.edgesy = [MESH.edgesy; [mesh{block}.edgesy(:,1:2)+mesh{block-1}
                .nn_g_cs mesh{block}.edgesy(:,3:4)]];
            MESH.volumes = [MESH.volumes; [mesh{block}.volumes(:,1:2)+mesh{block
                -1}.ne_g_h_cs mesh{block}.volumes(:,3:4)+mesh{block-1}.ne_g_v_cs
                mesh{block}.volumes(:,5:6)]];
        else
            mesh{block}.nn_g_cs = nn_g;
            mesh{block}.ne_g_cs = ne_g;
            mesh{block}.ne_g_h_cs = ne_g_h;
            mesh{block}.ne_g_v_cs = ne_g_v;
            mesh{block}.nv_g_cs = nv_g;
            MESH.edgesx = [MESH.edgesx; mesh{block}.edgesx];
            MESH.edgesy = [MESH.edgesy; mesh{block}.edgesy];
            MESH.volumes = [MESH.volumes; mesh{block}.volumes];
        end

        %% Boundaries per block
        % Initialize
        bc{block} = boundaries(mesh{block});

        % Create coordinates matrix used to remove duplicates
        if block==2
            tmp = mesh{block}.XY;
            tmp((bc{block}.l_n & ~bc{block}.t_n),1) = tmp((bc{block}.l_n & ~bc{
                block}.t_n),1)+h/10;
%             tmp(bc{block}.l_n,1) = tmp(bc{block}.l_n,1)+h/10;
            MESH.XYt = [MESH.XYt; tmp]; clear tmp;
        elseif block==6
            tmp = mesh{block}.XY;
            tmp((bc{block}.l_n & ~bc{block}.b_n),1) = tmp((bc{block}.l_n & ~bc{
                block}.b_n),1)+h/10;
%             tmp(bc{block}.l_n,1) = tmp(bc{block}.l_n,1)+h/10;
            MESH.XYt = [MESH.XYt; tmp]; clear tmp;
        else
            MESH.XYt = [MESH.XYt; mesh{block}.XY];
        end

        % Block unit normal function
        n1{block} = @(x,y) -(x==min(domain.x))+(x==max(domain.x));
        n2{block} = @(x,y) -(y==min(domain.y))+(y==max(domain.y));
    end
end
```

```
prec = -floor(log(h))+5;
clearvars -regexp tmp[01][01][xy]$ ^n[en][^{Deriv}]*
disp('Mesh blocks done')

%% Remove duplicates
% Create indices
if diffr
    [~,ian,icn] = unique(round(MESH.XYt,prec),'rows','stable');
else
    [~,ian,icn] = unique(round(MESH.XY,prec),'rows','stable');
end
MESH.XYt = MESH.XYt(ian,:); %round(MESH.XYt(ian,:),prec);
MESH.XY = MESH.XY(ian,:); %round(MESH.XY(ian,:),prec);
MESH.X = MESH.X(ian,:); %round(MESH.X(ian),prec);
MESH.Y = MESH.Y(ian,:); %round(MESH.Y(ian),prec);
MESH.XYold = MESH.XY;
MESH.edgesx(:,1:2) = icn(MESH.edgesx(:,1:2));
MESH.edgesx(:,3) = (MESH.XY(MESH.edgesx(:,1),1)+MESH.XY(MESH.edgesx(:,2),1))/2;
MESH.edgesx(:,4) = (MESH.XY(MESH.edgesx(:,1),2)+MESH.XY(MESH.edgesx(:,2),2))/2;
MESH.edgesy(:,1:2) = icn(MESH.edgesy(:,1:2));
MESH.edgesy(:,3) = (MESH.XY(MESH.edgesy(:,1),1)+MESH.XY(MESH.edgesy(:,2),1))/2;
MESH.edgesy(:,4) = (MESH.XY(MESH.edgesy(:,1),2)+MESH.XY(MESH.edgesy(:,2),2))/2;
[~,iaex,icex] = unique(round(MESH.edgesx,prec),'rows','stable');
MESH.edgesx = MESH.edgesx(iaex,:); % MESH.edgesx(:,3:4) = round(MESH.edgesx
    (:,3:4),prec);
[~,iaey,icey] = unique(round(MESH.edgesy,prec),'rows','stable');
MESH.edgesy = MESH.edgesy(iaey,:); % MESH.edgesy(:,3:4) = round(MESH.edgesy
    (:,3:4),prec);
MESH.volumes(:,1:4) = [icex(MESH.volumes(:,1:2)) icey(MESH.volumes(:,3:4))];
MESH.volumes(:,5:6) = round(MESH.volumes(:,5:6),prec);
MESH.nn_g = max(icn);
MESH.ne_g = max(icex)+max(icey);
MESH.ne_g_h = max(icex);
MESH.ne_g_v = max(icey);
MESH.nv_g = mesh{6}.nv_g_cs;
MESH.nvar.n = MESH.nn_g*(nDeriv+1)^2;
MESH.nvar.e_h = MESH.ne_g_h+nDeriv*(MESH.nn_g*2+MESH.ne_g_h);
MESH.nvar.e_v = MESH.ne_g_v+nDeriv*(MESH.nn_g*2+MESH.ne_g_v);
MESH.nvar.e = MESH.ne_g+nDeriv*(MESH.nn_g*4+MESH.ne_g);
MESH.nvar.v = MESH.nv_g+nDeriv*(MESH.nn_g+MESH.ne_g);

%% Reorder mesh points
[MESH.XYt,ind_n] = sortrows(MESH.XYt,2);
ind_n = [ind_n zeros(MESH.nn_g,1)];
for i=1:MESH.nn_g
    ind_n(i,2) = find(ind_n(:,1)==i);
end
icn = ind_n(icn,2);
MESH.XY = MESH.XY(ind_n(:,1),:);
MESH.X = MESH.X(ind_n(:,1),:);
MESH.Y = MESH.Y(ind_n(:,1),:);
MESH.XYold = MESH.XYold(ind_n(:,1),:);
MESH.edgesx(:,1:2) = [ind_n(MESH.edgesx(:,1),2) ind_n(MESH.edgesx(:,2),2)];
MESH.edgesy(:,1:2) = [ind_n(MESH.edgesy(:,1),2) ind_n(MESH.edgesy(:,2),2)];
[MESH.edgesx,ind_ex] = sortrows(MESH.edgesx,4);
ind_ex = [ind_ex zeros(MESH.ne_g_h,1)];
for i=1:MESH.ne_g_h
```

```
    ind_ex(i,2) = find(ind_ex(:,1)==i);
end
icex = ind_ex(icex,2);
[MESH.edgesy,ind_ey] = sortrows(MESH.edgesy,4);
ind_ey = [ind_ey zeros(MESH.ne_g_v,1)];
for i=1:MESH.ne_g_v
    ind_ey(i,2) = find(ind_ey(:,1)==i);
end
icey = ind_ey(icey,2);
MESH.volumes(:,1:4) = [ind_ex(MESH.volumes(:,1),2) ind_ex(MESH.volumes(:,2),2)
    ind_ey(MESH.volumes(:,3),2) ind_ey(MESH.volumes(:,4),2)];
[MESH.volumes,ind_v] = sortrows(MESH.volumes,6);
ind_v = [ind_v zeros(MESH.nv_g,1)];
for i=1:MESH.nv_g
    ind_v(i,2) = find(ind_v(:,1)==i);
end
icv = ind_v(:,2);
for block=1:6
    mesh{block}.icn = icn((mesh{block}.nn_g_cs-mesh{block}.nn_g+1):mesh{block}
        .nn_g_cs);
    mesh{block}.icex = icex((mesh{block}.ne_g_h_cs-mesh{block}.ne_g_h+1):mesh{
        block}.ne_g_h_cs);
    mesh{block}.icey = icey((mesh{block}.ne_g_v_cs-mesh{block}.ne_g_v+1):mesh{
        block}.ne_g_v_cs);
end

% %% Plot mesh points
% data = MESH.XYt;
% figure(); scatter(data(:,1),data(:,2),'.'); hold on;
% c = cellstr(num2str((1:size(data,1))'));
% text(data(:,1)+(((data(:,1)-0.501)>0)*2-1.75)*0.02,data(:,2)+0.01,c);
% c = cellstr(num2str((1:size(MESH.edgesx,1))'));
% text(MESH.edgesx(:,3),MESH.edgesx(:,4),c);
% c = cellstr(num2str((1:size(MESH.edgesy,1))'));
% text(MESH.edgesy(:,3),MESH.edgesy(:,4),c);
% clear data;
disp('Duplicates and Reorder done')

%% Create Incidence Matrices
E10x1 = sparse(1:MESH.ne_g_h,MESH.edgesx(:,1),-1,MESH.ne_g_h,MESH.nn_g)+...
    sparse(1:MESH.ne_g_h,MESH.edgesx(:,2),1,MESH.ne_g_h,MESH.nn_g);
E10y1 = sparse(1:MESH.ne_g_v,MESH.edgesy(:,1),-1,MESH.ne_g_v,MESH.nn_g)+...
    sparse(1:MESH.ne_g_v,MESH.edgesy(:,2),1,MESH.ne_g_v,MESH.nn_g);
E21x1 = sparse(1:MESH.nv_g,MESH.volumes(:,1),-1,MESH.nv_g,MESH.ne_g_h)+...
    sparse(1:MESH.nv_g,MESH.volumes(:,2),1,MESH.nv_g,MESH.ne_g_h);
E21y1 = sparse(1:MESH.nv_g,MESH.volumes(:,3),1,MESH.nv_g,MESH.ne_g_v)+...
    sparse(1:MESH.nv_g,MESH.volumes(:,4),-1,MESH.nv_g,MESH.ne_g_v);
if nDeriv
    E10x = blkdiag(E10x1,sparse(1:MESH.nn_g,1:MESH.nn_g,1),E10x1,sparse(1:
        MESH.nn_g,1:MESH.nn_g,1));
    E10y = blkdiag(E10y1,E10y1,sparse(1:MESH.nn_g,1:MESH.nn_g,1),sparse(1:
        MESH.nn_g,1:MESH.nn_g,1));
    E21x = blkdiag(E21x1,E10y1,sparse(1:MESH.ne_g_h,1:MESH.ne_g_h,1),sparse(1:
        MESH.nn_g,1:MESH.nn_g,1));
    E21y = blkdiag(E21y1,-sparse(1:MESH.ne_g_v,1:MESH.ne_g_v,1),-E10x1,-sparse(1:
        MESH.nn_g,1:MESH.nn_g,1));
    E1d0dx = blkdiag(E10y1,E10y1,sparse(1:MESH.nn_g,1:MESH.nn_g,1),sparse(1:
```

```
        MESH.nn_g,1:MESH.nn_g,1)); % sparse((MESH.nn_g+MESH.ne_g_v)*2,MESH.nn_g
        *4,0); %
    E1d0dy = blkdiag(-E10x1,sparse(1:MESH.nn_g,1:MESH.nn_g,-1),-E10x1,sparse(1:
        MESH.nn_g,1:MESH.nn_g,-1)); % sparse((MESH.nn_g+MESH.ne_g_h)*2,MESH.nn_g
        *4,0); %
    E2d1dx = blkdiag(-E21y1,sparse(1:MESH.ne_g_v,1:MESH.ne_g_v,1),E10x1,sparse(1:
        MESH.nn_g,1:MESH.nn_g,1)); % E21y;Iey;-E10x;In
    E2d1dy = blkdiag(E21x1,E10y1,sparse(1:MESH.ne_g_h,1:MESH.ne_g_h,1),sparse(1:
        MESH.nn_g,1:MESH.nn_g,1)); % E21x;E10y;Iex;In
else
    E10x = E10x1;                                                              276
    E10y = E10y1;
    E21x = E21x1;
    E21y = E21y1;
    E1d0dx = E10y1;
    E1d0dy = -E10x1;                                                           281
    E2d1dx = -E21y1;
    E2d1dy = E21x1;
end
E10 = [E10x;E10y]; E1d0d = [E1d0dx;E1d0dy];
E21 = [E21x E21y]; E2d1d = [E2d1dx E2d1dy];                                    286
% E21*E10
% E2d1d*E1d0d
clearvars -regexp E[12][01][\w*] E[12]d[01]d[\w*]
E10x = E10(1:MESH.nvar.e_h,:);
E10y = E10(MESH.nvar.e_h+1:end,:);                                             291
disp('Incidence matrices done')

%% Block to domain Mass Matrices
M00 = blkdiag(M00{1:6});
M11x = blkdiag(M11x{1:6});                                                     296
M11y = blkdiag(M11y{1:6});
M2d2d = blkdiag(M2d2d{1:6});
M02d = blkdiag(M02d{1:6});
M11dxy = blkdiag(M11dxy{1:6});
M11dyx = blkdiag(M11dyx{1:6});                                                 301
M1d1dx = blkdiag(M1d1dx{1:6});
M1d1dy = blkdiag(M1d1dy{1:6});

icn1 = cell(6,1); icex1 = icn1; icey1 = icn1; icv1 = icn1;
for block=1:6                                                                  306
    if block>1
        icn1{block} = icn(mesh{block-1}.nn_g_cs+1:mesh{block}.nn_g_cs);
        icex1{block} = icex(mesh{block-1}.ne_g_h_cs+1:mesh{block}.ne_g_h_cs);
        icey1{block} = icey(mesh{block-1}.ne_g_v_cs+1:mesh{block}.ne_g_v_cs);
        icv1{block} = icv(mesh{block-1}.nv_g_cs+1:mesh{block}.nv_g_cs);        311
        if nDeriv
            ic00 = [ic00; icn1{block}; icn1{block}+MESH.nn_g; icn1{block}+
                MESH.nn_g*2; icn1{block}+MESH.nn_g*3];
            ic11x = [ic11x; icex1{block}; icn1{block}+MESH.ne_g_h; icex1{block}+
                MESH.nn_g+MESH.ne_g_h; icn1{block}+MESH.nn_g+MESH.ne_g_h*2];
            ic11y = [ic11y; icey1{block}; icey1{block}+MESH.ne_g_v; icn1{block}+
                MESH.ne_g_v*2; icn1{block}+MESH.nn_g+MESH.ne_g_v*2];
            ic22 = [ic22; icv1{block}; icey1{block}+MESH.nv_g; icex1{block}+   316
                MESH.nv_g+MESH.ne_g_v; icn1{block}+MESH.nv_g+MESH.ne_g];
        end
    else
```

```
        icn1{block} = icn(1:mesh{block}.nn_g_cs);
        icex1{block} = icex(1:mesh{block}.ne_g_h_cs);
        icey1{block} = icey(1:mesh{block}.ne_g_v_cs);
        icv1{block} = icv(1:mesh{block}.nv_g_cs);
        if nDeriv
            ic00 = [icn1{block}; icn1{block}+MESH.nn_g; icn1{block}+MESH.nn_g*2;
                icn1{block}+MESH.nn_g*3];
            ic11x = [icex1{block}; icn1{block}+MESH.ne_g_h; icex1{block}+
                MESH.nn_g+MESH.ne_g_h; icn1{block}+MESH.nn_g+MESH.ne_g_h*2];
            ic11y = [icey1{block}; icey1{block}+MESH.ne_g_v; icn1{block}+
                MESH.ne_g_v*2; icn1{block}+MESH.nn_g+MESH.ne_g_v*2];
            ic22 = [icv1{block}; icey1{block}+MESH.nv_g; icex1{block}+MESH.nv_g+
                MESH.ne_g_v; icn1{block}+MESH.nv_g+MESH.ne_g];
        end
    end
end
if ~nDeriv
    ic00 = icn;
    ic11x = icex;
    ic11y = icey;
    ic22 = icv;
end
[a,b,c] = find(M00); M00 = sparse(ic00(a),ic00(b),c);
[a,b,c] = find(M11x); M11x = sparse(ic11x(a),ic11x(b),c);
[a,b,c] = find(M11y); M11y = sparse(ic11y(a),ic11y(b),c);
M11 = blkdiag(M11x,M11y);
[a,b,c] = find(M2d2d); M2d2d = sparse(ic22(a),ic22(b),c);
[a,b,c] = find(M02d); M02d = sparse(ic00(a),ic22(b),c);
[a,b,c] = find(M11dxy); M11dxy = sparse(ic11x(a),ic11y(b),c);
[a,b,c] = find(M11dyx); M11dyx = sparse(ic11y(a),ic11x(b),c);
M11d = blkdiag(M11dxy,M11dyx);
[a,b,c] = find(M1d1dx); M1d1dx = sparse(ic11x(a),ic11x(b),c);
[a,b,c] = find(M1d1dy); M1d1dy = sparse(ic11y(a),ic11y(b),c);
M1d1d = blkdiag(M1d1dy,M1d1dx);
clearvars a b c block*
disp('Block to domain done')

%% Find boundary indices
for i=1:6
    BC.bc_r_n{i} = false(MESH.nn_g,1); BC.bc_r_n{i}(icn1{i}) = bc{i}.r_n;
    BC.bc_l_n{i} = false(MESH.nn_g,1); BC.bc_l_n{i}(icn1{i}) = bc{i}.l_n;
    BC.bc_t_n{i} = false(MESH.nn_g,1); BC.bc_t_n{i}(icn1{i}) = bc{i}.t_n;
    BC.bc_b_n{i} = false(MESH.nn_g,1); BC.bc_b_n{i}(icn1{i}) = bc{i}.b_n;
    BC.bc_r_e{i} = false(MESH.ne_g_v,1); BC.bc_r_e{i}(icey1{i}) = bc{i}.r_e;
    BC.bc_l_e{i} = false(MESH.ne_g_v,1); BC.bc_l_e{i}(icey1{i}) = bc{i}.l_e;
    BC.bc_t_e{i} = false(MESH.ne_g_h,1); BC.bc_t_e{i}(icex1{i}) = bc{i}.t_e;
    BC.bc_b_e{i} = false(MESH.ne_g_h,1); BC.bc_b_e{i}(icex1{i}) = bc{i}.b_e;
end
BC.l_n = (BC.bc_l_n{1} | BC.bc_l_n{3} | BC.bc_l_n{5}); BC.l_e = (BC.bc_l_e{1} |
    BC.bc_l_e{3} | BC.bc_l_e{5});
BC.r_n = (BC.bc_r_n{2} | BC.bc_r_n{4} | BC.bc_r_n{6}); BC.r_e = (BC.bc_r_e{2} |
    BC.bc_r_e{4} | BC.bc_r_e{6});
BC.t_n = (BC.bc_t_n{5} | BC.bc_t_n{6}); BC.t_e = (BC.bc_t_e{5} | BC.bc_t_e{6});
BC.b_n = (BC.bc_b_n{1} | BC.bc_b_n{2}); BC.b_e = (BC.bc_b_e{1} | BC.bc_b_e{2});
BC.tl_n = BC.bc_t_n{5}; BC.tl_e = BC.bc_t_e{5}; BC.tr_n = BC.bc_t_n{6}; BC.tr_e =
     BC.bc_t_e{6};
BC.bl_n = BC.bc_b_n{1}; BC.bl_e = BC.bc_b_e{1}; BC.br_n = BC.bc_b_n{2}; BC.br_e =
```

```
    BC.bc_b_e{2};
BC.fl_tl_n = BC.bc_r_n{5}; BC.fl_tl_e = BC.bc_r_e{5};
BC.fl_tr_n = BC.bc_l_n{6}; BC.fl_tr_e = BC.bc_l_e{6};
BC.fl_ml_n = BC.bc_r_n{3}; BC.fl_ml_e = BC.bc_r_e{3};
BC.fl_mr_n = BC.bc_l_n{4}; BC.fl_mr_e = BC.bc_l_e{4};
BC.fl_bl_n = BC.bc_r_n{1}; BC.fl_bl_e = BC.bc_r_e{1};
BC.fl_br_n = BC.bc_l_n{2}; BC.fl_br_e = BC.bc_l_e{2};
BC.fl_l_n = (BC.fl_tl_n | BC.fl_bl_n); BC.fl_r_n = (BC.fl_tr_n | BC.fl_br_n);
BC.fl_n = (BC.fl_tl_n | BC.fl_tr_n | BC.fl_bl_n | BC.fl_br_n);
BC.fl_l_e = (BC.fl_tl_e | BC.fl_bl_e); BC.fl_r_e = (BC.fl_tr_e | BC.fl_br_e);
BC.fl_e = (BC.fl_tl_e | BC.fl_tr_e | BC.fl_bl_e | BC.fl_br_e);
% BC.l1_n = (abs(MESH.X-domain.xs(1))<10^-prec & (MESH.Y-domain.ys(2))<10^-prec);
% BC.l1_e = (abs(MESH.edgesy(:,3)-domain.xs(1))<10^-prec & (MESH.edgesy(:,4)-
    domain.ys(2))<10^-prec);
% BC.l2_n = (abs(MESH.X-domain.xs(1))<10^-prec & (MESH.Y-domain.ys(3))<10^-prec &
     (domain.ys(2)-MESH.Y)<10^-prec);
% BC.l2_e = (abs(MESH.edgesy(:,3)-domain.xs(1))<10^-prec & (MESH.edgesy(:,4)-
    domain.ys(3))<10^-prec & (domain.ys(2)-MESH.edgesy(:,4))<10^-prec);
% BC.l3_n = (abs(MESH.X-domain.xs(1))<10^-prec & (domain.ys(3)-MESH.Y)<10^-prec);
% BC.l3_e = (abs(MESH.edgesy(:,3)-domain.xs(1))<10^-prec & (domain.ys(3)-
    MESH.edgesy(:,4))<10^-prec);
% BC.l_n = abs(MESH.X-domain.xs(1))<10^-prec;
% BC.l_e = abs(MESH.edgesy(:,3)-domain.xs(1))<10^-prec;
% BC.r1_n = (abs(MESH.X-domain.xs(3))<10^-prec & (MESH.Y-domain.ys(2))<10^-prec);
% BC.r1_e = (abs(MESH.edgesy(:,3)-domain.xs(3))<10^-prec & (MESH.edgesy(:,4)-
    domain.ys(2))<10^-prec);
% BC.r2_n = (abs(MESH.X-domain.xs(3))<10^-prec & (MESH.Y-domain.ys(3))<10^-prec &
     (domain.ys(2)-MESH.Y)<10^-prec);
% BC.r2_e = (abs(MESH.edgesy(:,3)-domain.xs(3))<10^-prec & (MESH.edgesy(:,4)-
    domain.ys(3))<10^-prec & (domain.ys(2)-MESH.edgesy(:,4))<10^-prec);
% BC.r3_n = (abs(MESH.X-domain.xs(3))<10^-prec & (domain.ys(3)-MESH.Y)<10^-prec);
% BC.r3_e = (abs(MESH.edgesy(:,3)-domain.xs(3))<10^-prec & (domain.ys(3)-
    MESH.edgesy(:,4))<10^-prec);
% BC.r_n = abs(MESH.X-domain.xs(3))<10^-prec;
% BC.r_e = abs(MESH.edgesy(:,3)-domain.xs(3))<10^-prec;
% BC.b_n = abs(MESH.Y-domain.ys(1))<10^-prec;
% BC.b_e = abs(MESH.edgesx(:,4)-domain.ys(1))<10^-prec;
% BC.t_n = abs(MESH.Y-domain.ys(4))<10^-prec;
% BC.t_e = abs(MESH.edgesx(:,4)-domain.ys(4))<10^-prec;
% if diffr
%     BC.bl_n = ((MESH.XYt(:,1)<=domain.xs(2) & MESH.Y==domain.ys(1)));
%     BC.bl_e = ((MESH.edgesx(:,3)<=domain.xs(2) & MESH.edgesx(:,4)==domain.ys(1)
    ));
%     BC.br_n = ((MESH.XYt(:,1)>domain.xs(2) & MESH.Y==domain.ys(1)));
%     BC.br_e = ((MESH.edgesx(:,3)>domain.xs(2) & MESH.edgesx(:,4)==domain.ys(1))
    );
%     BC.tl_n = (((MESH.XYt(:,1)-domain.xs(2))<10^-prec & abs(MESH.Y-domain.ys(4)
    )<10^-prec));
%     BC.tl_e = (((MESH.edgesx(:,3)-domain.xs(2))<10^-prec & abs(MESH.edgesx(:,4)
    -domain.ys(4))<10^-prec));
%     BC.tr_n = ((MESH.XYt(:,1)>domain.xs(2) & abs(MESH.Y-domain.ys(4))<10^-prec)
    );
%     BC.tr_e = ((MESH.edgesx(:,3)>domain.xs(2) & abs(MESH.edgesx(:,4)-domain.ys
    (4))<10^-prec));
%     BC.fl_tl_n = (abs(MESH.XYt(:,1)-domain.xs(2))<10^-prec & (domain.ys(3)-
    MESH.Y)<10^-prec);
%     BC.fl_tl_e = (abs(MESH.edgesy(:,3)-domain.xs(2))<10^-prec & (domain.ys(3)-
```

```
    MESH.edgesy(:,4))<10^-prec);
%     BC.fl_tr_n = (MESH.XYt(:,1)>domain.xs(2)+h/20 & MESH.XYt(:,1)<domain.xs(2)+
    h/5 & (domain.ys(3)-MESH.Y)<10^-prec);
%     BC.fl_br_n = (MESH.XYt(:,1)>domain.xs(2)+h/20 & MESH.XYt(:,1)<domain.xs(2)+
    h/5 & (MESH.Y-domain.ys(2))<10^-prec);
%     BC.fl_tr_e = (MESH.edgesy(:,3)>domain.xs(2)+h/25 & MESH.edgesy(:,3)<
    domain.xs(2)+h/5 & MESH.edgesy(:,4)>=domain.ys(3));
%     BC.fl_bl_n = (abs(MESH.XYt(:,1)-domain.xs(2))<10^-prec & (MESH.Y-domain.ys
    (2))<10^-prec);
%     BC.fl_bl_e = (abs(MESH.edgesy(:,3)-domain.xs(2))<10^-prec & (MESH.edgesy
    (:,4)-domain.ys(2))<10^-prec);
%     BC.fl_br_e = (MESH.edgesy(:,3)>domain.xs(2)+h/25 & MESH.edgesy(:,3)<
    domain.xs(2)+h/5 & MESH.edgesy(:,4)<=domain.ys(2));
%     BC.fl_ml_n = (MESH.XYt(:,1)==domain.xs(2) & MESH.Y<=domain.ys(3) & MESH.Y>=
    domain.ys(2));
%     BC.fl_ml_e = (MESH.edgesy(:,3)==domain.xs(2) & MESH.edgesy(:,4)<=domain.ys
    (3) & MESH.edgesy(:,4)>=domain.ys(2));
%     BC.fl_mr_n = (MESH.XYt(:,1)>domain.xs(2)+h/20 & MESH.XYt(:,1)<domain.xs(2)+
    h/5 & (MESH.Y-domain.ys(3))<10^-prec & (domain.ys(2)-MESH.Y)<10^-prec);
%     BC.fl_mr_e = (MESH.edgesy(:,3)>domain.xs(2)+h/25 & MESH.edgesy(:,3)<
    domain.xs(2)+h/5 & MESH.edgesy(:,4)<=domain.ys(3) & MESH.edgesy(:,4)>=
    domain.ys(2));
%     BC.fl_l_n = (BC.fl_tl_n | BC.fl_bl_n); BC.fl_r_n = (BC.fl_tr_n | BC.fl_br_n
    );
%     BC.fl_n = (BC.fl_tl_n | BC.fl_tr_n | BC.fl_bl_n | BC.fl_br_n);
%     BC.fl_l_e = (BC.fl_tl_e | BC.fl_bl_e); BC.fl_r_e = (BC.fl_tr_e | BC.fl_br_e
    );
%     BC.fl_e = (BC.fl_tl_e | BC.fl_tr_e | BC.fl_bl_e | BC.fl_br_e);
% end

%% Create boundary Mass matrices
N1 = cell(6,1); N2=N1; tmp1y=N1; tmp1x=N1; tmp2y=N1; tmp2x=N1;
for block=1:6
    [m00_BC.r{block},m11_BC.r{block},m01_BC.r{block}] =
        mass_matrices_2D_v4_2_BCtest(mesh{block},0,1,MESH);
    [m00_BC.l{block},m11_BC.l{block},m01_BC.l{block}] =
        mass_matrices_2D_v4_2_BCtest(mesh{block},0,-1,MESH);
    [m00_BC.b{block},m11_BC.b{block},m01_BC.b{block}] =
        mass_matrices_2D_v4_2_BCtest(mesh{block},-1,0,MESH);
    [m00_BC.t{block},m11_BC.t{block},m01_BC.t{block}] =
        mass_matrices_2D_v4_2_BCtest(mesh{block},1,0,MESH);
end
M00_BC.l = matplus(m00_BC.l,[1 3 5]); M11_BC.l = matplus(m11_BC.l,[1 3 5]);
    M01_BC.l = matplus(m01_BC.l,[1 3 5]);
M00_BC.r = matplus(m00_BC.r,[2 4 6]); M11_BC.r = matplus(m11_BC.r,[2 4 6]);
    M01_BC.r = matplus(m01_BC.r,[2 4 6]);
if diffr
    M00_BC.tl = m00_BC.t{5}; M11_BC.tl = m11_BC.t{5}; M01_BC.tl = m01_BC.t{5};
    M00_BC.tr = m00_BC.t{6}; M11_BC.tr = m11_BC.t{6}; M01_BC.tr = m01_BC.t{6};
    M00_BC.bl = m00_BC.b{1}; M11_BC.bl = m11_BC.b{1}; M01_BC.bl = m01_BC.b{1};
    M00_BC.br = m00_BC.b{2}; M11_BC.br = m11_BC.b{2}; M01_BC.br = m01_BC.b{2};
    M00_BC.t = M00_BC.tl+M00_BC.tr; M11_BC.t = M11_BC.tl+M11_BC.tr; M01_BC.t =
        M01_BC.tl+M01_BC.tr;
    M00_BC.b = M00_BC.bl+M00_BC.br; M11_BC.b = M11_BC.bl+M11_BC.br; M01_BC.b =
        M01_BC.bl+M01_BC.br;
    M00_BC.fl_ml = m00_BC.r{3}; M11_BC.fl_ml = m11_BC.r{3}; M01_BC.fl_ml =
        m01_BC.r{3};
```

```
    M00_BC.fl_mr = m00_BC.l{4}; M11_BC.fl_mr = m11_BC.l{4}; M01_BC.fl_mr =
        m01_BC.l{4};
    M00_BC.fl_l = matplus(m00_BC.r,[1 5]); M11_BC.fl_l = matplus(m11_BC.r,[1 5]);
         M01_BC.fl_l = matplus(m01_BC.r,[1 5]);
    M00_BC.fl_r = matplus(m00_BC.l,[2 6]); M11_BC.fl_r = matplus(m11_BC.l,[2 6]);
         M01_BC.fl_r = matplus(m01_BC.l,[2 6]);
    M00_BC.fl = M00_BC.fl_l+M00_BC.fl_r; M11_BC.fl = M11_BC.fl_l+M11_BC.fl_r;
        M01_BC.fl = M01_BC.fl_l+M01_BC.fl_r;
else
    M00_BC.t = matplus(m00_BC.t,[5 6]); M11_BC.t = matplus(m11_BC.t,[5 6]);
        M01_BC.t = matplus(m01_BC.t,[5 6]);
    M00_BC.b = matplus(m00_BC.b,[1 2]); M11_BC.b = matplus(m11_BC.b,[1 2]);
        M01_BC.b = matplus(m01_BC.b,[1 2]);
end

%% Normals
if diffr
    n1 = @(x,y) -(abs(x-domain.xs(1))<10^-prec) + (abs(x-domain.xs(2))<10^-prec)
        - (x>domain.xs(2)+h/20 & x<domain.xs(2)+h/5) + (abs(x-domain.xs(3))<10^-
        prec);
    n1_1 = @(x,y) -(abs(x-domain.xs(1))<10^-prec) - (abs(x-domain.xs(2))<10^-prec
        ) + (abs(x-domain.xs(3))<10^-prec);
else
    n1 = @(x,y) -(abs(x-domain.xs(1))<10^-prec)+(abs(x-domain.xs(3))<10^-prec);
end
n2 = @(x,y) -(abs(y-domain.ys(1))<10^-prec)+(abs(y-domain.ys(4))<10^-prec);

nx.be = sparse(MESH.nvar.e_v,1,0); ny.be = sparse(-BC.b_e);
nx.te = sparse(MESH.nvar.e_v,1,0); ny.te = sparse(+BC.t_e);
nx.le = sparse(-BC.l_e); ny.le = sparse(MESH.nvar.e_h,1,0);
nx.re = sparse(+BC.r_e); ny.re = sparse(MESH.nvar.e_h,1,0);
nx.bn = sparse(MESH.nvar.n,1,0); ny.bn = sparse(-BC.b_n);
nx.tn = sparse(MESH.nvar.n,1,0); ny.tn = sparse(+BC.t_n);
nx.ln = sparse(-BC.l_n); ny.ln = sparse(MESH.nvar.n,1,0);
nx.rn = sparse(+BC.r_n); ny.rn = sparse(MESH.nvar.n,1,0);

if diffr
    nx.ble = sparse(MESH.nvar.e_v,1,0); ny.ble = sparse(-BC.bl_e);
    nx.bre = sparse(MESH.nvar.e_v,1,0); ny.bre = sparse(-BC.br_e);
    nx.tle = sparse(MESH.nvar.e_v,1,0); ny.tle = sparse(+BC.tl_e);
    nx.tre = sparse(MESH.nvar.e_v,1,0); ny.tre = sparse(+BC.tr_e);
    nx.bln = sparse(MESH.nvar.n,1,0); ny.bln = sparse(-BC.bl_n);
    nx.brn = sparse(MESH.nvar.n,1,0); ny.brn = sparse(-BC.br_n);
    nx.tln = sparse(MESH.nvar.n,1,0); ny.tln = sparse(+BC.tl_n);
    nx.trn = sparse(MESH.nvar.n,1,0); ny.trn = sparse(+BC.tr_n);
    nx.fl_ble = sparse(+BC.fl_bl_e); ny.fl_ble = sparse(MESH.nvar.e_h,1,0);
    nx.fl_mle = sparse(+BC.fl_ml_e); ny.fl_mle = sparse(MESH.nvar.e_h,1,0);
    nx.fl_tle = sparse(+BC.fl_tl_e); ny.fl_tle = sparse(MESH.nvar.e_h,1,0);
    nx.fl_bre = sparse(-BC.fl_br_e); ny.fl_bre = sparse(MESH.nvar.e_h,1,0);
    nx.fl_mre = sparse(-BC.fl_mr_e); ny.fl_mre = sparse(MESH.nvar.e_h,1,0);
    nx.fl_tre = sparse(-BC.fl_tr_e); ny.fl_tre = sparse(MESH.nvar.e_h,1,0);
    nx.fl_bln = sparse(+BC.fl_bl_n); ny.fl_bln = sparse(MESH.nvar.n,1,0);
    nx.fl_mln = sparse(+BC.fl_ml_n); ny.fl_mln = sparse(MESH.nvar.n,1,0);
    nx.fl_tln = sparse(+BC.fl_tl_n); ny.fl_tln = sparse(MESH.nvar.n,1,0);
    nx.fl_brn = sparse(-BC.fl_br_n); ny.fl_brn = sparse(MESH.nvar.n,1,0);
    nx.fl_mrn = sparse(-BC.fl_mr_n); ny.fl_mrn = sparse(MESH.nvar.n,1,0);
    nx.fl_trn = sparse(-BC.fl_tr_n); ny.fl_trn = sparse(MESH.nvar.n,1,0);
```

```
    nx.fl_le = nx.fl_ble+nx.fl_tle; nx.fl_ln = nx.fl_bln+nx.fl_tln;
    nx.fl_re = nx.fl_bre+nx.fl_tre; nx.fl_rn = nx.fl_brn+nx.fl_trn;

    if nDeriv                                                              496
        tmp.nx_le = [nx.le;nx.le;nx.ln;nx.ln];
        tmp.nx_re = [nx.re;nx.re;nx.rn;nx.rn];
        tmp.ny_te = [ny.te;ny.tn;ny.te;ny.tn];
        tmp.ny_be = [ny.be;ny.bn;ny.be;ny.bn];
        tmp.nx_ln = [nx.ln;nx.ln;nx.ln;nx.ln];                             501
        tmp.nx_rn = [nx.rn;nx.rn;nx.rn;nx.rn];
        tmp.ny_tn = [ny.tn;ny.tn;ny.tn;ny.tn];
        tmp.ny_bn = [ny.bn;ny.bn;ny.bn;ny.bn];
        tmp.nx_fl_le = [nx.fl_le;nx.fl_le;nx.fl_ln;nx.fl_ln];
        tmp.nx_fl_ln = [nx.fl_ln;nx.fl_ln;nx.fl_ln;nx.fl_ln];              506
        tmp.nx_fl_re = [nx.fl_re;nx.fl_re;nx.fl_rn;nx.fl_rn];
        tmp.nx_fl_rn = [nx.fl_rn;nx.fl_rn;nx.fl_rn;nx.fl_rn];
        tmp.ny_ble = [ny.ble;ny.bln;ny.ble;ny.bln];
        tmp.ny_bln = [ny.bln;ny.bln;ny.bln;ny.bln];
        tmp.ny_bre = [ny.bre;ny.brn;ny.bre;ny.brn];                        511
        tmp.ny_brn = [ny.brn;ny.brn;ny.brn;ny.brn];
        tmp.ny_tle = [ny.tle;ny.tln;ny.tle;ny.tln];
        tmp.ny_tln = [ny.tln;ny.tln;ny.tln;ny.tln];
        tmp.ny_tre = [ny.tre;ny.trn;ny.tre;ny.trn];
        tmp.ny_trn = [ny.trn;ny.trn;ny.trn;ny.trn];                        516

        nx.le = tmp.nx_le; nx.re = tmp.nx_re;
        ny.te = tmp.ny_te; ny.be = tmp.ny_be;
        nx.ln = tmp.nx_ln; nx.rn = tmp.nx_rn;
        ny.tn = tmp.ny_tn; ny.bn = tmp.ny_bn;                              521
        nx.fl_le = tmp.nx_fl_le; nx.fl_ln = tmp.nx_fl_ln;
        nx.fl_re = tmp.nx_fl_re; nx.fl_rn = tmp.nx_fl_rn;
        ny.ble = tmp.ny_ble; ny.bln = tmp.ny_bln;
        ny.bre = tmp.ny_bre; ny.brn = tmp.ny_brn;
        ny.tle = tmp.ny_tle; ny.tln = tmp.ny_tln;                          526
        ny.tre = tmp.ny_tre; ny.trn = tmp.ny_trn; clear tmp;
    end
end
disp('Boundaries done')
                                                                           531
%% Exact solution and boundary function
Fun.PHI = @(x,y) exp(1i*k*(x*cos(theta)+y*sin(theta)));
Fun.PHIpx = @(x,y) 1i*k*cos(theta)*Fun.PHI(x,y);
Fun.PHIpy = @(x,y) 1i*k*sin(theta)*Fun.PHI(x,y);
Fun.PHIpxy = @(x,y) -k^2*cos(theta)*sin(theta)*Fun.PHI(x,y);               536
if ~plane_wave && ~diffr
    Fun.PHI = @(x,y) sin(k*x).*sin(k*y);
    Fun.PHIpx = @(x,y) k*cos(k*x).*sin(k*y);
    Fun.PHIpy = @(x,y) k*sin(k*x).*cos(k*y);
    Fun.PHIpxy = @(x,y) k^2*cos(k*x).*cos(k*y);                            541
end
Fun.Ux = @(x,y) -Fun.PHI(x,y)*cos(theta);
Fun.Uxpx = @(x,y) -Fun.PHIpx(x,y)*cos(theta);
Fun.Uxpy = @(x,y) -Fun.PHIpy(x,y)*cos(theta);
Fun.Uxpxy = @(x,y) -Fun.PHIpxy(x,y)*cos(theta);                            546
Fun.Uy = @(x,y) -Fun.PHI(x,y)*sin(theta);
Fun.Uypx = @(x,y) -Fun.PHIpx(x,y)*sin(theta);
```

```
Fun.Uypy = @(x,y) -Fun.PHIpy(x,y)*sin(theta);
Fun.Uypxy = @(x,y) -Fun.PHIpxy(x,y)*sin(theta);
g = @(x,y) ( 1i*k .* (Fun.PHI(x,y)) .* ( cos(theta).*n1(x,y)+sin(theta).*n2(x,y)
    - 1 ) );
gx = @(x,y) ( 1i*k .* (Fun.PHI(x,y)) .* ( cos(theta).*n1(x,y) - 1 ) );
gy = @(x,y) ( 1i*k .* (Fun.PHI(x,y)) .* ( sin(theta).*n2(x,y) - 1 ) );
gt = @(x,y,xt,yt) ( 1i*k .* (Fun.PHI(x,y)) .* ( cos(theta).*n1(xt,yt)+sin(theta).
    *n2(x,y) - 1 ) );
gxt = @(x,y,xt,yt) ( 1i*k .* (Fun.PHI(x,y)) .* ( cos(theta).*n1(xt,yt) - 1 ) );
if nDeriv
    % g
    gpx = @(x,y) ( 1i*k .* (Fun.PHIpx(x,y)) .* ( cos(theta).*n1(x,y)+sin(theta).*
        n2(x,y) - 1 ) );
    gpy = @(x,y) ( 1i*k .* (Fun.PHIpy(x,y)) .* ( cos(theta).*n1(x,y)+sin(theta).*
        n2(x,y) - 1 ) );
    gpxy = @(x,y) ( 1i*k .* (Fun.PHIpxy(x,y)) .* ( cos(theta).*n1(x,y)+sin(theta)
        .*n2(x,y) - 1 ) );
    % gx
    gxpx = @(x,y) ( 1i*k .* (Fun.PHIpx(x,y)) .* ( cos(theta).*n1(x,y) - 1 ) );
    gxpy = @(x,y) ( 1i*k .* (Fun.PHIpy(x,y)) .* ( cos(theta).*n1(x,y) - 1 ) );
    gxpxy = @(x,y) ( 1i*k .* (Fun.PHIpxy(x,y)) .* ( cos(theta).*n1(x,y) - 1 ) );
    % gt
    gpxt = @(x,y,xt,yt) ( 1i*k .* (Fun.PHIpx(x,y)) .* ( cos(theta).*n1(xt,yt)+sin
        (theta).*n2(x,y) - 1 ) );
    gpyt = @(x,y,xt,yt) ( 1i*k .* (Fun.PHIpy(x,y)) .* ( cos(theta).*n1(xt,yt)+sin
        (theta).*n2(x,y) - 1 ) );
    gpxyt = @(x,y,xt,yt) ( 1i*k .* (Fun.PHIpxy(x,y)) .* ( cos(theta).*n1(xt,yt)+
        sin(theta).*n2(x,y) - 1 ) );
    % gxt
    gxpxt = @(x,y,xt,yt) ( 1i*k .* (Fun.PHIpx(x,y)) .* ( cos(theta).*n1(xt,yt) -
        1 ) );
    gxpyt = @(x,y,xt,yt) ( 1i*k .* (Fun.PHIpy(x,y)) .* ( cos(theta).*n1(xt,yt) -
        1 ) );
    gxpxyt = @(x,y,xt,yt) ( 1i*k .* (Fun.PHIpxy(x,y)) .* ( cos(theta).*n1(xt,yt)
        - 1 ) );
    % gy
    gypx = @(x,y) ( 1i*k .* (Fun.PHIpx(x,y)) .* ( sin(theta).*n2(x,y) - 1 ) );
    gypy = @(x,y) ( 1i*k .* (Fun.PHIpy(x,y)) .* ( sin(theta).*n2(x,y) - 1 ) );
    gypxy = @(x,y) ( 1i*k .* (Fun.PHIpxy(x,y)) .* ( sin(theta).*n2(x,y) - 1 ) );
    % Assemble
    g = @(x,y) [g(x,y);gpx(x,y);gpy(x,y);gpxy(x,y)];
    gx = @(x,y) [gx(x,y);gxpx(x,y);gxpy(x,y);gxpxy(x,y)];
    gy = @(x,y) [gy(x,y);gypx(x,y);gypy(x,y);gypxy(x,y)];
    gt = @(x,y,xt,yt) [gt(x,y,xt,yt);gpxt(x,y,xt,yt);gpyt(x,y,xt,yt);gpxyt(x,y,xt
        ,yt)];
    gxt = @(x,y,xt,yt) [gxt(x,y,xt,yt);gxpxt(x,y,xt,yt);gxpyt(x,y,xt,yt);gxpxyt(x
        ,y,xt,yt)];
end

%% External forcing
Fun.F = @(x,y) 0.*x.*y;
Fun.Fpx = Fun.F;
Fun.Fpy = Fun.F;
Fun.Fpxy = Fun.F;
if ~plane_wave && ~diffr
    Fun.F = @(x,y) (2+gam)*k^2/(1i*k)*Fun.PHI(x,y);
    Fun.Fpx = @(x,y) (2+gam)*k^2/(1i*k)*Fun.PHIpx(x,y);
```

```
    Fun.Fpy = @(x,y) (2+gam)*k^2/(1i*k)*Fun.PHIpy(x,y);
    Fun.Fpxy = @(x,y) (2+gam)*k^2/(1i*k)*Fun.PHIpxy(x,y);
elseif plane_wave && ~diffr
    Fun.F = @(x,y) (1+gam)*k^2/(1i*k)*Fun.PHI(x,y);
    Fun.Fpx = @(x,y) (1+gam)*k^2/(1i*k)*Fun.PHIpx(x,y);
    Fun.Fpy = @(x,y) (1+gam)*k^2/(1i*k)*Fun.PHIpy(x,y);
    Fun.Fpxy = @(x,y) (1+gam)*k^2/(1i*k)*Fun.PHIpxy(x,y);
end
f = @(x,y) Fun.F(x,y);
if nDeriv>0
    fpx = @(x,y) Fun.Fpx(x,y);
    fpy = @(x,y) Fun.Fpy(x,y);
    fpxy = @(x,y) Fun.Fpxy(x,y);
    f = @(x,y) [f(x,y);fpx(x,y);fpy(x,y);fpxy(x,y)];
end
disp('Exact and external forcing done')

%% Integrate U
if integr_U
    ux_i = cell(6,1); uy_i = ux_i; uxpx_i = ux_i; uxpy_i = ux_i; uxpxy_i = ux_i;
        uypx_i = ux_i; uypy_i = ux_i; uypxy_i = ux_i;
    for block=1:1:6
        ux_i{block,1} = interpolate2D_1form_y_v2(Fun.Ux,mesh{block},mesh{block}.P
            +1,1);
        uy_i{block,1} = interpolate2D_1form_x_v2(Fun.Uy,mesh{block},mesh{block}.P
            +1,1);
        if nDeriv
            uxpx_i{block,1} = interpolate2D_1form_y_v2(Fun.Uxpx,mesh{block},mesh{
                block}.P+1,1);
            uxpy_i{block,1} = Fun.Ux(mesh{block}.XGrid,mesh{block}.YGrid);
            uxpxy_i{block,1} = Fun.Uxpx(mesh{block}.XGrid,mesh{block}.YGrid);
            uypx_i{block,1} = Fun.Uy(mesh{block}.XGrid,mesh{block}.YGrid);
            uypy_i{block,1} = interpolate2D_1form_x_v2(Fun.Uypy,mesh{block},mesh{
                block}.P+1,1);
            uypxy_i{block,1} = Fun.Uypy(mesh{block}.XGrid,mesh{block}.YGrid);
        end
    end
    Ux_l = [ux_i{1}(:,1);ux_i{3}(:,1);ux_i{5}(:,1)];
    Uy_b = transpose([uy_i{1}(1,:) uy_i{2}(1,:)]);
    if diffr
        Uy_tl = transpose(uy_i{5}(1,:));
        Uy_tl1 = zeros(MESH.nvar.e,1); Uy_tl1(BC.tl_e+MESH.ne_g_v,1) = Uy_tl;
        Uy_bl = transpose(uy_i{1}(1,:));
        Uy_bl1 = zeros(MESH.nvar.e,1); Uy_bl1(BC.bl_e+MESH.ne_g_v,1) = Uy_bl;
    end
    Ux_l1 = zeros(MESH.nvar.e,1);
    Uy_b1 = zeros(MESH.nvar.e,1);
    if nDeriv
        Ux_l = [Ux_l;uxpx_i{1}(:,1);uxpx_i{3}(:,1);uxpx_i{5}(:,1);uxpy_i{1}(1:end
            -1,1);uxpy_i{3}(1:end-1,1);uxpy_i{5}(:,1);uxpxy_i{1}(1:end-1,1);
            uxpxy_i{3}(1:end-1,1);uxpxy_i{5}(:,1)];
        Uy_b = [Uy_b;transp([uypx_i{1}(1,1:end-1) uypx_i{2}(1,:) uypy_i{1}(1,:)
            uypy_i{2}(1,:) uypxy_i{1}(1,1:end-1) uypxy_i{2}(1,:)])];
        Ux_l1([BC.l_e;BC.l_e+MESH.ne_g_v;BC.l_n+MESH.ne_g_v*2;BC.l_n+MESH.ne_g_v
            *2+MESH.nn_g],1) = Ux_l;
        Uy_b1([BC.b_e;BC.b_n+MESH.ne_g_h;BC.b_e+MESH.ne_g_h+MESH.nn_g;BC.b_n+
            MESH.ne_g_h*2+MESH.nn_g]+MESH.nvar.e_v,1) = Uy_b;
```

```
    else
        Ux_l1(BC.l_e,1) = Ux_l;
        Uy_b1(BC.b_e+MESH.nvar.e_v,1) = Uy_b;
    end
end
display('Integrate U done')

%% Check symmetry
% sym.M = [check_sym(M00,MESH.XYt);
%     check_sym(M11x,MESH.edgesx(:,3:4));
%     check_sym(M11y,MESH.edgesy(:,3:4));
%     check_sym(M1d1dx,MESH.edgesx(:,3:4));
%     check_sym(M1d1dy,MESH.edgesy(:,3:4));
%     check_sym(M2d2d,MESH.volumes(:,5:6));
%     check_sym(M11dxy,MESH.edgesx(:,3:4),MESH.edgesy(:,3:4));
%     check_sym(M11dyx,MESH.edgesy(:,3:4),MESH.edgesx(:,3:4));
%     check_sym(M02d,MESH.XYt,MESH.volumes(:,5:6))];
% sym.E = [check_sym(E10x,MESH.edgesx(:,3:4),MESH.XYt);
%     check_sym(E10y,MESH.edgesy(:,3:4),MESH.XYt);
%     check_sym(E1d0d(1:MESH.nvar.e_v,:),MESH.edgesy(:,3:4),MESH.XYt);
%     check_sym(E1d0d(1+MESH.nvar.e_v:end,:),MESH.edgesx(:,3:4),MESH.XYt);
%     check_sym(E2d1d(:,1:MESH.nvar.e_v),MESH.volumes(:,5:6),MESH.edgesy(:,3:4));
%     check_sym(E2d1d(:,1+MESH.nvar.e_v:end),MESH.volumes(:,5:6),MESH.edgesx
    (:,3:4));
%     check_sym(E21(:,1:MESH.nvar.e_h),MESH.volumes(:,5:6),MESH.edgesx(:,3:4));
%     check_sym(E21(:,1+MESH.nvar.e_h:end),MESH.volumes(:,5:6),MESH.edgesy(:,3:4)
    )];
% names = {'l' 'r' 'fl_ml' 'fl_mr' 'fl_l' 'fl_r' 'fl'};
% for i = 1:numel(names)
%     sym.MBC(i,:) = [check_sym(M00_BC.(names{i}),MESH.XYt);
%         check_sym(M01_BC.(names{i})(:,1:MESH.nvar.e_v),MESH.XYt,MESH.edgesy
    (:,3:4));
%         check_sym(M01_BC.(names{i})(:,1+MESH.nvar.e_v:end),MESH.XYt,MESH.edgesx
    (:,3:4));
%         check_sym(M11_BC.(names{i})(1:MESH.nvar.e_v,1:MESH.nvar.e_v),
    MESH.edgesy(:,3:4));
%         check_sym(M11_BC.(names{i})(1+MESH.nvar.e_v:end,1+MESH.nvar.e_v:end),
    MESH.edgesx(:,3:4));];';
% end
% sym.MBC = [sym.MBC;
%     [check_sym(M00_BC.t+M00_BC.b,MESH.XYt);
%     check_sym(M01_BC.t(:,1:MESH.nvar.e_v)+M01_BC.b(:,1:MESH.nvar.e_v),MESH.XYt,
    MESH.edgesy(:,3:4));
%     check_sym(M01_BC.t(:,1+MESH.nvar.e_v:end)+M01_BC.b(:,1+MESH.nvar.e_v:end),
    MESH.XYt,MESH.edgesx(:,3:4));
%     check_sym(M11_BC.t(1:MESH.nvar.e_v,1:MESH.nvar.e_v)+M11_BC.b(1:
    MESH.nvar.e_v,1:MESH.nvar.e_v),MESH.edgesy(:,3:4));
%     check_sym(M11_BC.t(1+MESH.nvar.e_v:end,1+MESH.nvar.e_v:end)+M11_BC.b(1+
    MESH.nvar.e_v:end,1+MESH.nvar.e_v:end),MESH.edgesx(:,3:4));]'];

%% Construct Solution
if diffr % Diffraction model or square domain plane wave
    switch eqn
        case 0
            switch A
                case 1 % Interference
                    M00_BC.sum = M00_BC.l+M00_BC.r+M00_BC.fl;
```

```
                % 1-eqn
                lhs = -gam*k^2*M00 - E10'*M11*E10 + 1i*k*M00_BC.sum;
                rhs = M00*f(MESH.X,MESH.Y) - (M00_BC.l+M00_BC.fl_l)*gx(MESH.X
                    ,MESH.Y);

                % Periodic top bottom
                lhs(:,BC.t_n) = lhs(:,BC.t_n)+lhs(:,BC.b_n);
                lhs(:,BC.b_n) = []; lhs(BC.b_n,:) = []; rhs(BC.b_n,:) = [];

                SOL = lhs\rhs;
                phi = zeros(MESH.nvar.n,1); tmp = true(MESH.nvar.n,1); tmp(
                    BC.b_n) = false; phi(tmp,1) = SOL; clear tmp;
                phi(BC.b_n,1) = phi(BC.t_n,1);
            case 2 % Periodic left
                M00_BC.sum = M00_BC.l+M00_BC.r+M00_BC.tr+M00_BC.br+M00_BC.fl;

                % 1-eqn
                lhs = -gam*k^2*M00 - E10'*M11*E10 + 1i*k*M00_BC.sum;
                rhs = M00*f(MESH.X,MESH.Y) - (M00_BC.l+M00_BC.fl_l)*gx(MESH.X
                    ,MESH.Y);

                % Periodic top bottom
                lhs(:,BC.tl_n) = lhs(:,BC.tl_n)+lhs(:,BC.bl_n);
                lhs(:,BC.bl_n) = []; lhs(BC.bl_n,:) = []; rhs(BC.bl_n,:) =
                    [];

                SOL = lhs\rhs;
                phi = zeros(MESH.nvar.n,1); tmp = true(MESH.nvar.n,1); tmp(
                    BC.bl_n) = false; phi(tmp,1) = SOL; clear tmp;
                phi(BC.bl_n,1) = phi(BC.tl_n,1);
            case 3 % No periodic
                M00_BC.sum = M00_BC.l+M00_BC.r+M00_BC.t+M00_BC.b+M00_BC.fl_l;

                % 1-eqn
                lhs = -gam*k^2*M00 - E10'*M11*E10 + 1i*k*M00_BC.sum;
                rhs = M00*f(MESH.X,MESH.Y) - (M00_BC.l+M00_BC.fl_l)*gx(MESH.X
                    ,MESH.Y) - (M00_BC.bl+M00_BC.tl)*gy(MESH.X,MESH.Y) ;

                phi = lhs\rhs;
            case 4 % Prescribe U
        end
    case 1
        switch A
            case 1 % Interference
                M00_BC.sum = M00_BC.l+M00_BC.r+M00_BC.fl;
                % 2-eqn
                lhs1 = [ -gam*k^2*M00+1i*k*M00_BC.sum 1i*k*E10'*M11d ];
                lhs2 = [ M11d'*E10 1i*k*M1d1d ];
                rhs1 = M00*f(MESH.X,MESH.Y) - (M00_BC.l+M00_BC.fl_l)*gx(
                    MESH.X,MESH.Y);
                rhs2 = zeros(MESH.nvar.e,1);
                lhs = [lhs1;lhs2];
                rhs = [rhs1;rhs2];

                % Periodic top bottom
                if nDeriv
```

```
                    T = [BC.t_n;BC.t_n;BC.t_n;BC.t_n;false(MESH.nvar.e_v,1);
                        BC.t_e;BC.t_n;BC.t_e;BC.t_n];
                    B = [BC.b_n;BC.b_n;BC.b_n;BC.b_n;false(MESH.nvar.e_v,1);
                        BC.b_e;BC.b_n;BC.b_e;BC.b_n];
                else
                    Tn = [BC.t_n;false(MESH.nvar.e,1)];
                    Te = [false(MESH.nvar.n+MESH.nvar.e_v,1);BC.t_e];
                    Bn = [BC.b_n;false(MESH.nvar.e,1)];
                    Be = [false(MESH.nvar.n+MESH.nvar.e_v,1);BC.b_e];
                end
                lhs(:,Tn) = lhs(:,Tn)+(lhs(:,Bn));
                lhs(:,Te) = lhs(:,Te)+conj(lhs(:,Be));
                out = false(MESH.nvar.n+MESH.nvar.e,1);%[false(MESH.nvar.n,1)
                    ; BC.fl_r_e; false(MESH.nvar.e_h,1)];
                lhs(:,(Bn|Be|out)) = []; lhs((Bn|Be|out),:) = []; rhs((Bn|Be|
                    out),:) = [];

                SOL = lhs\rhs;
                sol = zeros(MESH.nvar.n+MESH.nvar.e,1);
                sol(~(Bn|Be|out)) = SOL;
                sol(Bn) = (sol(Tn));
                sol(Be) = conj(sol(Te));
                SOL=sol; clear B T sol
                phi = SOL(1:MESH.nvar.n); v = -E10*phi; vx = v(1:
                    MESH.nvar.e_h); vy = v(MESH.nvar.e_h+1:end);
                ux = SOL(MESH.nvar.n+1:(MESH.nvar.n+MESH.nvar.e_v));
                uy = SOL((MESH.nvar.n+MESH.nvar.e_v+1):end);
                psi = E2d1d*[ux;uy];
            case 2 % Periodic left
                M00_BC.sum = M00_BC.l+M00_BC.r+M00_BC.tr+M00_BC.br+M00_BC.fl;
                % 2-eqn
                lhs1 = [ -gam*k^2*M00+1i*k*M00_BC.sum 1i*k*E10'*M11d ];
%                 lhs1 = [ -gam*k^2*M00-1i*k*M00_BC.sum -1i*k*M02d*E2d1d-1i*k
    *((M01_BC.l+M01_BC.r+M01_BC.fl_l).*(ones(MESH.nvar.n,1)*([nx.le'+nx.re'+
    nx.fl_le' zeros(1,MESH.nvar.e_h)])) + (M01_BC.t+M01_BC.b).*(ones(MESH.nvar.n
    ,1)*([zeros(1,MESH.nvar.e_v) ny.tle'+ny.ble'+ny.tre'+ny.bre'])) ];
%                 lhs1 = [ -gam*k^2*M00-1i*k*M00_BC.sum -1i*k*M02d*E2d1d-1i*k
    *((M01_BC.l+M01_BC.r+M01_BC.fl_l).*(ones(MESH.nvar.n,1)*([nx.le'+nx.re'+
    nx.fl_le' zeros(1,MESH.nvar.e_h)])) + (M01_BC.t+M01_BC.b).*(ones(MESH.nvar.n
    ,1)*([zeros(1,MESH.nvar.e_v) ny.tle'+ny.ble'+ny.tre'+ny.bre'])) ];
                lhs2 = [ M11d'*E10 1i*k*M1d1d ];
                rhs1 = M00*f(MESH.X,MESH.Y) - (M00_BC.l+M00_BC.fl_l)*gx(
                    MESH.X,MESH.Y);
%                 rhs1 = M00*f(MESH.X,MESH.Y) + (M00_BC.l+M00_BC.fl_l)*gx(
    MESH.X,MESH.Y) + (M00_BC.bl+M00_BC.tl)*gy(MESH.X,MESH.Y);
%                 rhs1 = M00*f(MESH.X,MESH.Y) + (M00_BC.l+M00_BC.fl_l)*gx(
    MESH.X,MESH.Y);
                rhs2 = zeros(MESH.nvar.e,1);
                lhs = [lhs1;lhs2];
                rhs = [rhs1;rhs2];

                % Periodic top bottom
                if nDeriv
                    TL = [BC.tl_n;BC.tl_n;BC.tl_n;BC.tl_n;false(MESH.nvar.e_v
                        ,1);BC.tl_e;BC.tl_n;BC.tl_e;BC.tl_n];
                    BL = [BC.bl_n;BC.bl_n;BC.bl_n;BC.bl_n;false(MESH.nvar.e_v
                        ,1);BC.bl_e;BC.bl_n;BC.bl_e;BC.bl_n];
```

```
                else
                    TL = [BC.tl_n;false(MESH.nvar.e_v,1);BC.tl_e];
                    BL = [BC.bl_n;false(MESH.nvar.e_v,1);BC.bl_e];
                    TLn = [BC.tl_n;false(MESH.nvar.e,1)];
                    BLn = [BC.bl_n;false(MESH.nvar.e,1)];
                    TLe = [false(MESH.nvar.n+MESH.nvar.e_v,1);BC.tl_e];
                    BLe = [false(MESH.nvar.n+MESH.nvar.e_v,1);BC.bl_e];
                end
                TL = false(MESH.nvar.n+MESH.nvar.e,1);
                BL = false(MESH.nvar.n+MESH.nvar.e,1);
                lhs(:,TLn) = lhs(:,TLn)+lhs(:,BLn);
                lhs(:,TLe) = lhs(:,TLe)+conj(lhs(:,BLe));
%                 lhs(:,TL) = lhs(:,TL)+lhs(:,BL);
                out = false(MESH.nvar.n+MESH.nvar.e,1);%[false(MESH.nvar.n,1)
                    ; BC.fl_r_e; false(MESH.nvar.e_h,1)];
%                 out = false(MESH.nvar.n+MESH.nvar.e,1);
                lhs(:,BLn|BLe|out) = []; lhs(BLn|BLe|out,:) = []; rhs(BLn|BLe
                    |out,:) = [];

                SOL = lhs\rhs;
                sol = zeros(MESH.nvar.n+MESH.nvar.e,1);
                sol(~(BLn|BLe|out)) = SOL;
                sol(BLn) = sol(TLn);
                sol(BLe) = conj(sol(TLe));
                SOL=sol; clear BL TL sol
                phi = SOL(1:MESH.nvar.n); v = -E10*phi; vx = v(1:
                    MESH.nvar.e_h); vy = v(MESH.nvar.e_h+1:end);
                ux = SOL(MESH.nvar.n+1:(MESH.nvar.n+MESH.nvar.e_v));
                uy = SOL((MESH.nvar.n+MESH.nvar.e_v+1):end);
                psi = E2d1d*[ux;uy];
            case 3 % No periodic
                M00_BC.sum = M00_BC.l+M00_BC.r+M00_BC.t+M00_BC.b+M00_BC.fl;
                % 2-eqn
                rx.e = cos(atan((MESH.edgesy(:,4)-mean(domain.ys))./(
                    MESH.edgesy(:,3)-domain.xs(2)))).*sign(MESH.edgesy(:,3)-
                    domain.xs(2));
                ry.e = sin(atan((MESH.edgesx(:,4)-mean(domain.ys))./(
                    MESH.edgesx(:,3)-domain.xs(2)))).*sign(MESH.edgesx(:,3)-
                    domain.xs(2));
                rx.n = cos(atan((MESH.Y-mean(domain.ys))./(MESH.X-domain.xs
                    (2)))).*sign(MESH.X-domain.xs(2));
                ry.n = sin(atan((MESH.Y-mean(domain.ys))./(MESH.X-domain.xs
                    (2)))).*sign(MESH.X-domain.xs(2));
                bx = {'l','r','fl_l'}; by = {'tl','tr','bl','br','t','b'};
                for i=1:numel(bx)
                    rx.([bx{i},'e']) = rx.e; rx.([bx{i},'e'])(~BC.([bx{i},'_e
                        '])) = 0;
                    rx.([bx{i},'n']) = rx.n; rx.([bx{i},'n'])(~BC.([bx{i},'_n
                        '])) = 0;
                    ry.([bx{i},'e']) = zeros(size(ry.e)); ry.([bx{i},'n']) =
                        zeros(size(ry.n));
                end
                for i=1:numel(by)
                    ry.([by{i},'e']) = ry.e; ry.([by{i},'e'])(~BC.([by{i},'_e
                        '])) = 0;
                    ry.([by{i},'n']) = ry.n; ry.([by{i},'n'])(~BC.([by{i},'_n
                        '])) = 0;
```

```
                    rx.([by{i},'e']) = zeros(size(rx.e)); rx.([by{i},'n']) =
                        zeros(size(rx.n));
                end
%                   lhs1 = [ -gam*k^2*M00-1i*k*M00_BC.sum -1i*k*M02d*E2d1d-1i*k
    *((M01_BC.l+M01_BC.r+M01_BC.fl_l).*(ones(MESH.nvar.n,1)*([nx.le'+nx.re'+
    nx.fl_le' zeros(size(ry.re'))])) + (M01_BC.t+M01_BC.b).*(ones(MESH.nvar.n,1)
    *([zeros(size(nx.tle'+nx.ble'+rx.tre'+rx.bre')) ny.tle'+ny.ble'+ny.tre'+ny.bre
    '])))  ];
%                   lhs2 = [ M11d'*E10 1i*k*M1d1d ];
%                   rhs1 = M00*f(MESH.X,MESH.Y) + (M00_BC.l+M00_BC.fl_l)*gx(
    MESH.X,MESH.Y) + (M00_BC.bl+M00_BC.tl)*gy(MESH.X,MESH.Y);
%                   rhs2 = zeros(MESH.nvar.e,1);
                lhs1 = [ -gam*k^2*M00+1i*k*M00_BC.sum 1i*k*E10'*M11d ];
                lhs2 = [ M11d'*E10 1i*k*M1d1d ];
                rhs1 = M00*f(MESH.X,MESH.Y) - (M00_BC.l+M00_BC.fl_l)*gx(
                    MESH.X,MESH.Y) - (M00_BC.bl+M00_BC.tl)*gy(MESH.X,MESH.Y);
                rhs2 = zeros(MESH.nvar.e,1);

                lhs = [lhs1;lhs2];
                rhs = [rhs1;rhs2];
                out = false(MESH.nvar.n+MESH.nvar.e,1);%[false(MESH.nvar.n,1)
                    ; BC.fl_r_e; false(MESH.nvar.e_h,1)];
                lhs(out,:) = []; lhs(:,out) = []; rhs(out,:) = [];
                SOL = lhs\rhs;
                sol = zeros(MESH.nvar.n+MESH.nvar.e,1); sol(~out) = SOL; SOL
                    = sol; clear sol;
                phi = SOL(1:MESH.nvar.n); v = -E10*phi; vx = v(1:
                    MESH.nvar.e_h); vy = v(MESH.nvar.e_h+1:end);
                ux = SOL(MESH.nvar.n+1:(MESH.nvar.n+MESH.nvar.e_v));
                uy = SOL((MESH.nvar.n+MESH.nvar.e_v+1):end);
                psi = E2d1d*[ux;uy];
            case 4 % Prescribe U
        end
    case 2
        switch A
            case 1 % Interference
                bx_w = {'fl_r'}; by_w = {}; bx_g = {'l','fl_l'}; by_g = {};
                    bx_iog = {'l','fl_l','r'}; by_iog = {};
                bper = {'t','b'}; bU = {};
            case 2 % Periodic left
                bx_w = {'fl_r'}; by_w = {}; bx_g = {'l','fl_l'}; by_g = {};
                    bx_iog = {'l','fl_l','r'}; by_iog = {'br','tr'};
                bper = {'tl','bl'}; bU = {};
            case 3 % No periodic
%                   bx_w = {'fl_r'}; by_w = {}; bx_g = {'l','fl_l'}; by_g = {'
    bl','tl'}; bx_iog = {'l','fl_l','r'}; by_iog = {'b','t'};
                bx_w = {'fl_l','fl_r'}; by_w = {}; bx_g = {'l'}; by_g = {'bl'
                    ,'tl'}; bx_iog = {'l','r'}; by_iog = {'b','t'};
                bper = {}; bU = {};
            case 4 % Prescribe U
                bx_g = {'fl_l'}; by_g = {'bl','tl'}; bx_iog = {'fl_l','r'};
                    by_iog = {'b','t'};
                bper = {}; bU = {'l'};
        end

        %% Set Boundaries
        M00_BC.iogx = sparse(MESH.nvar.n,MESH.nvar.n,0); M01_BC.iogx = sparse
```

```
    (MESH.nvar.n,MESH.nvar.e,0); M11_BC.iogx = sparse(MESH.nvar.e,
    MESH.nvar.e,0);
M00_BC.iogy = sparse(MESH.nvar.n,MESH.nvar.n,0); M01_BC.iogy = sparse
    (MESH.nvar.n,MESH.nvar.e,0); M11_BC.iogy = sparse(MESH.nvar.e,
    MESH.nvar.e,0);
M00_BC.gx = M00_BC.iogx; M01_BC.gx = M01_BC.iogx; M11_BC.gx =
    M11_BC.iogx; M00_BC.gy = M00_BC.iogy; M01_BC.gy = M01_BC.iogy;
    M11_BC.gy = M11_BC.iogy;
M00_BC.wx = M00_BC.iogx; M01_BC.wx = M01_BC.iogx; M11_BC.wx =
    M11_BC.iogx; M00_BC.wy = M00_BC.iogy; M01_BC.wy = M01_BC.iogy;
    M11_BC.gy = M11_BC.iogy;
nx.ioge = sparse(MESH.nvar.e_v,1,0); nx.iogn = sparse(MESH.nvar.n
    ,1,0); ny.ioge = sparse(MESH.nvar.e_h,1,0); ny.iogn = sparse(
    MESH.nvar.n,1,0);
nx.ge = nx.ioge; nx.gn = nx.iogn; ny.ge = ny.ioge; ny.gn = ny.iogn;
nx.we = nx.ioge; nx.wn = nx.iogn; ny.we = ny.ioge; ny.wn = ny.iogn;
for i=1:numel(bx_iog)
    M00_BC.iogx = M00_BC.iogx+M00_BC.(bx_iog{i}); M01_BC.iogx =
        M01_BC.iogx+M01_BC.(bx_iog{i}); M11_BC.iogx = M11_BC.iogx+
        M11_BC.(bx_iog{i});
    nx.ioge = nx.ioge + nx.([bx_iog{i},'e']); nx.iogn = nx.iogn + nx.
        ([bx_iog{i},'n']);
end
nx.iogn(abs(nx.iogn)>1) = nx.iogn(abs(nx.iogn)>1)./abs(nx.iogn(abs(
    nx.iogn)>1));
for i=1:numel(by_iog)
    M00_BC.iogy = M00_BC.iogy+M00_BC.(by_iog{i}); M01_BC.iogy =
        M01_BC.iogy+M01_BC.(by_iog{i}); M11_BC.iogy = M11_BC.iogy+
        M11_BC.(by_iog{i});
    ny.ioge = ny.ioge + ny.([by_iog{i},'e']); ny.iogn = ny.iogn + ny.
        ([by_iog{i},'n']);
end
for i=1:numel(bx_g)
    M00_BC.gx = M00_BC.gx+M00_BC.(bx_g{i}); M01_BC.gx = M01_BC.gx+
        M01_BC.(bx_g{i}); M11_BC.gx = M11_BC.gx+M11_BC.(bx_g{i});
    nx.ge = nx.ge + nx.([bx_g{i},'e']); nx.gn = nx.gn + nx.([bx_g{i},
        'n']);
end
nx.gn(abs(nx.gn)>1) = nx.gn(abs(nx.gn)>1)./abs(nx.gn(abs(nx.gn)>1));
for i=1:numel(by_g)
    M00_BC.gy = M00_BC.gy+M00_BC.(by_g{i}); M01_BC.gy = M01_BC.gy+
        M01_BC.(by_g{i}); M11_BC.gy = M11_BC.gy+M11_BC.(by_g{i});
    ny.ge = ny.ge + ny.([by_g{i},'e']); ny.gn = ny.gn + ny.([by_g{i},
        'n']);
end
for i=1:numel(bx_w)
    M00_BC.wx = M00_BC.wx+M00_BC.(bx_w{i}); M01_BC.wx = M01_BC.wx+
        M01_BC.(bx_w{i}); M11_BC.wx = M11_BC.wx+M11_BC.(bx_w{i});
    nx.we = nx.we + nx.([bx_w{i},'e']); nx.wn = nx.wn + nx.([bx_w{i},
        'n']);

end
nx.wn(abs(nx.wn)>1) = nx.wn(abs(nx.wn)>1)./abs(nx.wn(abs(nx.wn)>1));
for i=1:numel(by_w)
    M00_BC.wy = M00_BC.wy+M00_BC.(by_w{i}); M01_BC.wy = M01_BC.wy+
        M01_BC.(by_w{i}); M11_BC.wy = M11_BC.wy+M11_BC.(by_w{i});
    ny.we = ny.we + ny.([by_w{i},'e']); ny.wn = ny.wn + ny.([by_w{i},
```

```
            'n']);
        end

        %% LHS
        Re_phi_phi = k^2*M00 + E10'*M11*E10 + k^2*(M00_BC.iogx+M00_BC.iogy);
        Re_phi_u = k^2*( M01_BC.iogx.*(ones(MESH.nvar.n,1)*[nx.ioge' zeros(1,
            MESH.nvar.e_h)]) + ...
            M01_BC.iogy.*(ones(MESH.nvar.n,1)*[zeros(1,MESH.nvar.e_v) ny.ioge
                ']) );
        Im_phi_phi = sparse(MESH.nvar.n,MESH.nvar.n,0);
        Im_phi_u = k*(E10'*M11d+gam*M02d*E2d1d);
        Re_u_u = k^2*M1d1d + E2d1d'*M2d2d*E2d1d + ...
            k^2*(([nx.ioge;zeros(MESH.nvar.e_h,1)]*[nx.ioge' zeros(1,
                MESH.nvar.e_h)]).*M11_BC.iogx + ...
            ([zeros(MESH.nvar.e_v,1);ny.ioge]*[zeros(1,MESH.nvar.e_v) ny.ioge
                ']).*M11_BC.iogy);
        Re_u_phi = k^2*( M01_BC.iogx'.*([nx.ioge;zeros(MESH.nvar.e_h,1)]*ones
            (1,MESH.nvar.n)) + ...
            M01_BC.iogy'.*([zeros(MESH.nvar.e_v,1);ny.ioge]*ones(1,
                MESH.nvar.n)) );
        Im_u_u = sparse(MESH.nvar.e,MESH.nvar.e,0);
        Im_u_phi = -k*(M11d'*E10+gam*E2d1d'*M02d');%-k*gam*( M01_BC.wx'.*([
            nx.we;zeros(MESH.nvar.e_h,1)]*ones(1,MESH.nvar.n)) + M01_BC.wy'.
            *([zeros(MESH.nvar.e_v,1);ny.we]*ones(1,MESH.nvar.n)) );
        lhs1 = [Re_phi_phi -Im_phi_phi Re_phi_u -Im_phi_u];
        lhs2 = [Im_phi_phi Re_phi_phi Im_phi_u Re_phi_u];
        lhs3 = [Re_u_phi -Im_u_phi Re_u_u -Im_u_u];
        lhs4 = [Im_u_phi Re_u_phi Im_u_u Re_u_u];
        lhs = [lhs1;lhs2;lhs3;lhs4]; clearvars -regexp lhs[?] Re_* Im_*

        %% RHS
        Re_phi_rhs = -gam*M00*f(MESH.X,MESH.Y);
        Im_phi_rhs = k*( M00_BC.gx*gxt(MESH.X,MESH.Y,MESH.XYt(:,1),MESH.XYt
            (:,2))+M00_BC.gy*gy(MESH.X,MESH.Y) );
        Re_u_rhs = sparse(MESH.nvar.e,1);
        Im_u_rhs = 1/k*E2d1d'*M02d'*f(MESH.X,MESH.Y) + ...
            k*( M01_BC.gx'.*([nx.ge;zeros(MESH.nvar.e_h,1)]*ones(1,
                MESH.nvar.n))*gxt(MESH.X,MESH.Y,MESH.XYt(:,1),MESH.XYt(:,2)) +
                 ...
            M01_BC.gy'.*([zeros(MESH.nvar.e_v,1);ny.ge]*ones(1,MESH.nvar.n))*
                gy(MESH.X,MESH.Y) );
        rhs1 = real(Re_phi_rhs) - imag(Im_phi_rhs);
        rhs2 = real(Re_phi_rhs) + real(Im_phi_rhs);
        rhs3 = real(Re_u_rhs) - imag(Im_u_rhs);
        rhs4 = imag(Re_u_rhs) + real(Im_u_rhs);
        rhs = [rhs1;rhs2;rhs3;rhs4]; clearvars -regexp rhs[?] Re_* Im_*

%           %% Prescribed U
%           if isempty(bU)
%               indU = false(MESH.nvar.n*2+MESH.nvar.e*2,1);
%           else
%               if nDeriv
%                   indU = [false(MESH.nvar.n*2,1);BC.([bper{1},'_e']);BC.([
    bper{1},'_n']);BC.([bper{1},'_e']);BC.([bper{1},'_n']);false(MESH.nvar.e_v,1);
    BC.([bper{1},'_e']);BC.([bper{1},'_n']);BC.([bper{1},'_e']);BC.([bper{1},'_n
    '])];
%               else
```

```
%                     indU = [false(MESH.nvar.n*2,1);BC.([bU{1},'_e']);false(
    MESH.nvar.e_h,1);BC.([bU{1},'_e']);false(MESH.nvar.e_h,1)];
%                 end
%             end
% %             (bU{1})
%             ind_U = [BC.([bU{1},'_e']);]                                    941
%                 ind_tblr = [BC.bl_e;BC.tl_e]+MESH.nvar.n*2+MESH.nvar.e_v;
    ind_tbli = ind_tblr+MESH.nvar.e;
%                 ind_out = [ind_lr;ind_li;ind_tblr;ind_tbli];
%                 ind_out_v = [real(Ux_l);imag(Ux_l);real(Uy_bl);real(Uy_tl);imag
    (Uy_bl);imag(Uy_tl)];
%                 rhs = rhs-lhs(:,ind_out)*ind_out_v;
%                 SOL(ind_out) = ind_out_v;                                   946

            %% Periodicity | Walls
            PWx = false(MESH.nvar.n*2+MESH.nvar.e*2,1);
            for i=1:numel(bx_w)
                if nDeriv                                                   951
                    PWx = ( PWx | [false(MESH.nvar.n*2,1);BC.([bx_w{i},'_e']);
                        false(MESH.ne_g_v,1);false(MESH.nn_g*2,1);false(
                        MESH.nvar.e_h,1);BC.([bx_w{i},'_e']);false(MESH.ne_g_v,1);
                        false(MESH.nn_g*2,1);false(MESH.nvar.e_h,1)] );
                else
                    PWx = ( PWx | [false(MESH.nvar.n*2,1);BC.([bx_w{i},'_e']);
                        false(MESH.nvar.e_h,1);BC.([bx_w{i},'_e']);false(
                        MESH.nvar.e_h,1)] );
                end
            end                                                             956
            PWy = false(MESH.nvar.n*2+MESH.nvar.e*2,1);
            for i=1:numel(by_w)
                if nDeriv
                    PWy = ( PWy | [false(MESH.nvar.n*2,1);false(MESH.nvar.e_v,1);
                        BC.([by_w{i},'_e']);false(MESH.nvar.e_h-MESH.ne_g_h,1);
                        false(MESH.nvar.e_v,1);BC.([by_w{i},'_e']);false(
                        MESH.nvar.e_h-MESH.ne_g_h,1)] );
                else                                                        961
                    PWy = ( PWy | [false(MESH.nvar.n*2,1);false(MESH.nvar.e_v,1);
                        BC.([by_w{i},'_e']);false(MESH.nvar.e_v,1);BC.([by_w{i},'
                        _e'])] );
                end
            end
            P1 = false(MESH.nvar.n*2+MESH.nvar.e*2,1); P2 = P1;
            for i=1:size(bper,1)                                            966
                if nDeriv
                    P1 = ( P1 | [repmat(BC.([bper{i,1},'_n']),4,1);repmat(BC.([
                        bper{i,1},'_n']),4,1);false(MESH.nvar.e_v,1);BC.([bper{i
                        ,1},'_e']);BC.([bper{i,1},'_n']);BC.([bper{i,1},'_e']);BC.
                        ([bper{i,1},'_n']);false(MESH.nvar.e_v,1);BC.([bper{i,1},'
                        _e']);BC.([bper{i,1},'_n']);BC.([bper{i,1},'_e']);BC.([
                        bper{i,1},'_n'])] );
                    P2 = ( P2 | [repmat(BC.([bper{i,2},'_n']),4,1);repmat(BC.([
                        bper{i,2},'_n']),4,1);false(MESH.nvar.e_v,1);BC.([bper{i
                        ,2},'_e']);BC.([bper{i,2},'_n']);BC.([bper{i,2},'_e']);BC.
                        ([bper{i,2},'_n']);false(MESH.nvar.e_v,1);BC.([bper{i,2},'
                        _e']);BC.([bper{i,2},'_n']);BC.([bper{i,2},'_e']);BC.([
                        bper{i,2},'_n'])] );
                else
```

```
                P1 = ( P1 | [BC.([bper{i,1},'_n']);BC.([bper{i,1},'_n']);
                    false(MESH.nvar.e_v,1);BC.([bper{i,1},'_e']);false(
                    MESH.nvar.e_v,1);BC.([bper{i,1},'_e'])] );
                P2 = ( P2 | [BC.([bper{i,2},'_n']);BC.([bper{i,2},'_n']);
                    false(MESH.nvar.e_v,1);BC.([bper{i,2},'_e']);false(
                    MESH.nvar.e_v,1);BC.([bper{i,2},'_e'])] );
            end
        end

        lhs(:,P1) = lhs(:,P1)+lhs(:,P2);
        out = (P2 | PWx | PWy);
        lhs(:,out) = []; lhs(out,:) = []; rhs(out,:) = [];

        %% Calculate Solution
        SOL = lhs\rhs;
        sol = zeros(MESH.nvar.n*2+MESH.nvar.e*2,1);
        sol(~out) = SOL; sol(P2) = sol(P1); SOL=sol; clear P1 P2 PWx PWy sol
            out

        %% Collect Results
        phi = complex(SOL(1:MESH.nvar.n),SOL(MESH.nvar.n+1:MESH.nvar.n*2));
        ux = complex(SOL(MESH.nvar.n*2+1:(MESH.nvar.n*2+MESH.nvar.e_v)),SOL((
            MESH.nvar.n*2+MESH.nvar.e+1):(MESH.nvar.n*2+MESH.nvar.e+
            MESH.nvar.e_v)));
        uy = complex(SOL((MESH.nvar.n*2+MESH.nvar.e_v+1):(MESH.nvar.n*2+
            MESH.nvar.e)),SOL((MESH.nvar.n*2+MESH.nvar.e+MESH.nvar.e_v+1):end)
            );
        v = -E10*phi; vx = v(1:MESH.nvar.e_h); vy = v(MESH.nvar.e_h+1:end);
        psi = E2d1d*[ux;uy];

    end
    clearvars -regexp lhs[?] rhs[?] Re_* Im_*
else % no diffraction, no interference, no absorbing wall, open domain plane wave
    if plane_wave
        if ~integr_U % Model Problem A Demkowicz
            % Construct lhs
            Re_phi_phi = (E10'*M11*E10 + k^2.*M00 - (...
                +M01_BC.r(:,1:MESH.nvar.e_v)*(-E10y)*sin(theta)/(cos(theta))...
                +M01_BC.t(:,MESH.nvar.e_v+1:end)*(-E10x)*cos(theta)/(sin(theta)))
                    );
            %                       +M01_BC.l(:,1:MESH.nvar.e_v)*(-E10y)*sin(theta)
                /(-cos(theta))...
            %                           +M01_BC.b(:,MESH.nvar.e_v+1:end)*(-E10x)*
                cos(theta)/(-sin(theta))...
            Re_phi_u = sparse(MESH.nvar.n,MESH.nvar.e,0);
            Im_phi_phi = -k*(...
                +M00_BC.l...
                +M00_BC.b...
                +M00_BC.r/(cos(theta))...
                +M00_BC.t/(sin(theta)) );
            %                       +M00_BC.l/(-cos(theta))...
            %                           +M00_BC.b/(-sin(theta))...
            Im_phi_u = (gam/k-k)*M02d*E2d1d;
            Re_u_u = (E2d1d'*M2d2d*E2d1d+k^2*M1d1d);
            Re_u_phi = +M11_BC.r(:,1:MESH.nvar.e_v)*(-E10y)*sin(theta)...
                +M11_BC.t(:,MESH.nvar.e_v+1:end)*(-E10x)*cos(theta);
            %                       -M11_BC.l(:,1:MESH.nvar.e_v)*(-E10y)*sin(theta)
```

```
        ...
    %                           -M11_BC.b(:,MESH.nvar.e_v+1:end)*(-E10x)*
        cos(theta)...
    Im_u_u = k*(...
        +M11_BC.l.*([nx.le;ny.le]*[nx.le' ny.le'])...%*(-cos(theta))...
        +M11_BC.b.*([nx.be;ny.be]*[nx.be' ny.be'])...%*(-sin(theta))...
        +M11_BC.r.*([nx.re;ny.re]*[nx.re' ny.re'])*cos(theta)...
        +M11_BC.t.*blkdiag(nx.te*nx.te',(+ny.te)*(+ny.te)')*sin(theta) );
    Im_u_phi = (k-gam/k)*E2d1d'*M02d';
    lhs1 = [Re_phi_phi -Im_phi_phi Re_phi_u -Im_phi_u]; % HH: 1 HelmHoltz
         0 Reaction-Diffusion
    lhs2 = [Im_phi_phi Re_phi_phi Im_phi_u Re_phi_u];
    lhs3 = [Re_u_phi -Im_u_phi Re_u_u -Im_u_u];
    lhs4 = [Im_u_phi Re_u_phi Im_u_u Re_u_u];
    lhs = [lhs1;lhs2;lhs3;lhs4]; clearvars -regexp lhs[?] Re_* Im_*
    % Construct rhs
    Im_phi_rhs = gam/k*M00*ffp - k*( ...
        +M00_BC.b * gy(MESH.X,MESH.Y)...%/(-sin(theta))...
        +M00_BC.l * gx(MESH.X,MESH.Y) );%/(-cos(theta)) );
    Re_u_rhs = E2d1d'*M02d'*ffp;
    Im_u_rhs = k*( ...
        M01_BC.b' * gy(MESH.X,MESH.Y) .* [nx.be;ny.be] ...
        +M01_BC.l' * gx(MESH.X,MESH.Y) .*[nx.le;ny.le] );
    rhs1 = -imag(Im_phi_rhs);
    rhs2 = real(Im_phi_rhs);
    rhs3 = real(Re_u_rhs)-imag(Im_u_rhs);
    rhs4 = imag(Re_u_rhs)+real(Im_u_rhs);
    rhs = [rhs1;rhs2;rhs3;rhs4]; clearvars -regexp rhs[?] Re_* Im_*

    % Walls
    ind_walls = [];
    ind_out = ind_walls;
    lhs(:,ind_out) = []; lhs(ind_out,:) = []; rhs(ind_out,:) = []; clear
        tmp;
    display('Constructing Solution Done')
    % Calculate Solution
    sol = lhs\rhs;
    display('Solution Done')
    SOL = zeros((MESH.nvar.n+MESH.nvar.e)*2,1);
    tmp = false((MESH.nvar.n+MESH.nvar.e)*2,1); tmp(ind_out) = true;
    SOL(~tmp,1)=sol;
    sol=SOL;
else % Model Problem B Demkowicz
    % Construct lhs [real(phi)|imag(phi)|real(u)|imag(u)]
    display('Start Constructing Solution')
    lhs1 = [(E10'*M11*E10+k^2.*M00) k*(M00_BC.t+M00_BC.r) sparse(
        MESH.nvar.n,MESH.nvar.e,0) -(k*-M02d*E2d1d+gam/k*M02d*E2d1d)]; %
        HH: 1 HelmHoltz 0 Reaction-Diffusion
    lhs2 = [-k*(M00_BC.t+M00_BC.r) (E10'*M11*E10+k^2.*M00) (k*-M02d*E2d1d
        +gam/k*M02d*E2d1d) sparse(MESH.nvar.n,MESH.nvar.e,0)];
    lhs3 = [sparse(MESH.nvar.e,MESH.nvar.n,0) (k*(M01_BC.l'+M01_BC.b'-
        E2d1d'*M02d')+gam/k*E2d1d'*M02d') (E2d1d'*M2d2d*E2d1d+k^2*M1d1d) -
        k*(M11_BC.t+M11_BC.r)];
    lhs4 = [-(k*(M01_BC.l'+M01_BC.b'-E2d1d'*M02d')+gam/k*E2d1d'*M02d')
        sparse(MESH.nvar.e,MESH.nvar.n,0) k*(M11_BC.t+M11_BC.r) (E2d1d'*
        M2d2d*E2d1d+k^2*M1d1d)];
    lhs = [lhs1;lhs2;lhs3;lhs4];
```

```
            % Construct rhs
            rhs1 = -gam/k*M00*imag(ffp) + k*( M00_BC.t*imag(gy(MESH.X,MESH.Y)) +
                M00_BC.r*imag(gx(MESH.X,MESH.Y)) ) + k*( (M01_BC.l+M01_BC.b)*imag(
                Ux_l1+Uy_b1) );
            rhs2 = gam/k*M00*real(ffp) - k*( M00_BC.t*real(gy(MESH.X,MESH.Y)) +
                M00_BC.r*real(gx(MESH.X,MESH.Y)) ) - k*( (M01_BC.l+M01_BC.b)*real(
                Ux_l1+Uy_b1) );
            rhs3 = E2d1d'*M02d'*real(ffp) - k*( M01_BC.t'*imag(gy(MESH.X,MESH.Y))
                 + M01_BC.r'*imag(gx(MESH.X,MESH.Y)) );
            rhs4 = E2d1d'*M02d'*imag(ffp) + k*( M01_BC.t'*real(gy(MESH.X,MESH.Y))
                 + M01_BC.r'*real(gx(MESH.X,MESH.Y)) );
            rhs = [rhs1;rhs2;rhs3;rhs4];
            % Walls
            ind_walls = [];%[ind_l;ind_b];
            ind_out = ind_walls;
            lhs(:,ind_out) = []; lhs(ind_out,:) = []; rhs(ind_out,:) = [];
            display('Constructing Solution Done')
            % Calculate Solution
            sol = lhs\rhs;
            display('Solution Done')
            SOL = zeros((MESH.nvar.n+MESH.nvar.e)*2,1);
            tmp = false((MESH.nvar.n+MESH.nvar.e)*2,1); tmp(ind_out) = true;
            SOL(~tmp,1)=sol;
            sol=SOL; clear tmp;
        end
    else % sin()sin() model
        if 0 % Real
            % Construct lhs [real(phi)|imag(phi)|real(u)|imag(u)]
            display('Start Constructing Solution')
            lhs1 = [(E10'*M11*E10+omega^2.*M00) (-omega*M02d*E2d1d+gam/omega*M02d
                *E2d1d)]; % HH: 1 HelmHoltz 0 Reaction-Diffusion
            lhs2 = [(gam/omega*E2d1d'*M02d'-omega*E2d1d'*M02d') (E2d1d'*M2d2d*
                E2d1d+omega^2*M1d1d)];
            lhs = [lhs1;lhs2];
            % Construct rhs
            rhs1 = gam/omega*M00*real(ffp);
            rhs2 = E2d1d'*M02d'*real(ffp);
            rhs = [rhs1;rhs2];
            % Walls
            ind_walls = [BC.l_n;BC.r_n;BC.t_n;BC.b_n];
            ind_out = ind_walls;
            lhs(:,ind_out) = []; lhs(ind_out,:) = []; rhs(ind_out,:) = []; clear
                tmp;
            display('Constructing Solution Done')
            % Calculate Solution
            sol = lhs\rhs;
            display('Solution Done')
            SOL = zeros((MESH.nvar.n+MESH.nvar.e),1);
            tmp = false((MESH.nvar.n+MESH.nvar.e),1); tmp(ind_out) = true;
            SOL(~tmp,1)=sol; sol=SOL;
            SOL = zeros((MESH.nvar.n+MESH.nvar.e)*2,1);
            tmp = false((MESH.nvar.n+MESH.nvar.e)*2,1); tmp([(1:MESH.nvar.n)'+
                MESH.nvar.n;(1:MESH.nvar.e)'+MESH.nvar.n*2+MESH.nvar.e]) = true;
            SOL(~tmp,1)=sol;
            sol=SOL;
        else % Complex
            % Construct lhs [real(phi)|imag(phi)|real(u)|imag(u)]
```

```
display('Start Constructing Solution')
lhs1 = [(E10'*M11*E10+k^2.*M00) sparse(MESH.nvar.n,MESH.nvar.n,0)
    sparse(MESH.nvar.n,MESH.nvar.e,0) (k-gam/k)*M02d*E2d1d]; % HH: 1
    HelmHoltz 0 Reaction-Diffusion
lhs2 = [sparse(MESH.nvar.n,MESH.nvar.n,0) (E10'*M11*E10+k^2.*M00) -(k
    -gam/k)*M02d*E2d1d sparse(MESH.nvar.n,MESH.nvar.e,0)];
lhs3 = [sparse(MESH.nvar.e,MESH.nvar.n,0) -(k-gam/k)*E2d1d'*M02d' (
    E2d1d'*M2d2d*E2d1d+k^2*M1d1d) sparse(MESH.nvar.e,MESH.nvar.e,0)];
lhs4 = [(k-gam/k)*E2d1d'*M02d' sparse(MESH.nvar.e,MESH.nvar.n,0)
    sparse(MESH.nvar.e,MESH.nvar.e,0) (E2d1d'*M2d2d*E2d1d+k^2*M1d1d)];
lhs = [lhs1;lhs2;lhs3;lhs4];
% Construct rhs
rhs1 = -gam/k*M00*imag(ffp);
rhs2 = gam/k*M00*real(ffp);
rhs3 = E2d1d'*M02d'*real(ffp);
rhs4 = E2d1d'*M02d'*imag(ffp);
rhs = [rhs1;rhs2;rhs3;rhs4];
% Walls
tmp_n = [BC.l_n;BC.r_n;BC.t_n;BC.b_n];
if nDeriv
    ind_wallsn = [[tmp_n;tmp_n+MESH.nn_g;tmp_n+MESH.nn_g*2;tmp_n+
        MESH.nn_g*3];[tmp_n;tmp_n+MESH.nn_g;tmp_n+MESH.nn_g*2;tmp_n+
        MESH.nn_g*3]+MESH.nvar.n];
    rhs = rhs-lhs(:,tmp_n+MESH.nn_g)*real(Fun.PHIpx(MESH.X(tmp_n),
        MESH.Y(tmp_n)));
    rhs = rhs-lhs(:,tmp_n+MESH.nn_g*2)*real(Fun.PHIpy(MESH.X(tmp_n),
        MESH.Y(tmp_n)));
    rhs = rhs-lhs(:,tmp_n+MESH.nn_g*3)*real(Fun.PHIpxy(MESH.X(tmp_n),
        MESH.Y(tmp_n)));
    rhs = rhs-lhs(:,tmp_n+MESH.nn_g+MESH.nvar.n)*imag(Fun.PHIpx(
        MESH.X(tmp_n),MESH.Y(tmp_n)));
    rhs = rhs-lhs(:,tmp_n+MESH.nn_g*2+MESH.nvar.n)*imag(Fun.PHIpy(
        MESH.X(tmp_n),MESH.Y(tmp_n)));
    rhs = rhs-lhs(:,tmp_n+MESH.nn_g*3+MESH.nvar.n)*imag(Fun.PHIpxy(
        MESH.X(tmp_n),MESH.Y(tmp_n)));
else
    ind_wallsn = [tmp_n;tmp_n+MESH.nvar.n];
end
ind_out = ind_wallsn;
lhs(:,ind_out) = []; lhs(ind_out,:) = []; rhs(ind_out,:) = [];
display('Constructing Solution Done')
% Calculate Solution
sol = lhs\rhs;
display('Solution Done')
SOL = zeros((MESH.nvar.n+MESH.nvar.e)*2,1);
tmp = false((MESH.nvar.n+MESH.nvar.e)*2,1); tmp(ind_out) = true;
SOL(~tmp,1)=sol;
if nDeriv
    SOL(tmp_n+MESH.nn_g,1) = real(Fun.PHIpx(MESH.X(tmp_n),MESH.Y(
        tmp_n)));
    SOL(tmp_n+MESH.nn_g+MESH.nvar.n,1) = imag(Fun.PHIpx(MESH.X(tmp_n)
        ,MESH.Y(tmp_n)));
    SOL(tmp_n+MESH.nn_g*2,1) = real(Fun.PHIpy(MESH.X(tmp_n),MESH.Y(
        tmp_n)));
    SOL(tmp_n+MESH.nn_g*2+MESH.nvar.n,1) = imag(Fun.PHIpy(MESH.X(
        tmp_n),MESH.Y(tmp_n)));
    SOL(tmp_n+MESH.nn_g*3,1) = real(Fun.PHIpxy(MESH.X(tmp_n),MESH.Y(
```

```
                    tmp_n)));
                SOL(tmp_n+MESH.nn_g*3+MESH.nvar.n,1) = imag(Fun.PHIpxy(MESH.X(
                    tmp_n),MESH.Y(tmp_n)));
            end
            sol=SOL;                                                          1151
        end
    end
end

%% base set                                                                   1156
% Construct lhs [real(phi)|imag(phi)|real(u)|imag(u)]
% display('Start Constructing Solution')
% lhs1 = [(E10'*M11*E10+omega^2.*M00) sparse(MESH.nvar.n,MESH.nvar.n,0) sparse(
    MESH.nvar.n,MESH.nvar.e,0) -(omega*E10'*M11d+gam/omega*M02d*E2d1d)]; % HH: 1
    HelmHoltz 0 Reaction-Diffusion
% lhs2 = [sparse(MESH.nvar.n,MESH.nvar.n,0) (E10'*M11*E10+omega^2.*M00) (omega*
    E10'*M11d+gam/omega*M02d*E2d1d) sparse(MESH.nvar.n,MESH.nvar.e,0)];
% lhs3 = [sparse(MESH.nvar.e,MESH.nvar.n,0) (omega*M11d'*E10+gam/omega*E2d1d'*   1161
    M02d') (E2d1d'*M2d2d*E2d1d+omega^2*M1d1d) sparse(MESH.nvar.e,MESH.nvar.e,0)];
% lhs4 = [-(omega*M11d'*E10+gam/omega*E2d1d'*M02d') sparse(MESH.nvar.e,
    MESH.nvar.n,0) sparse(MESH.nvar.e,MESH.nvar.e,0) (E2d1d'*M2d2d*E2d1d+omega^2*
    M1d1d)];
% lhs = [lhs1;lhs2;lhs3;lhs4];
% % Construct rhs
% rhs1 = -gam/omega*M00*imag(ffp);
% rhs2 = gam/omega*M00*real(ffp);                                           1166
% rhs3 = E2d1d'*M02d'*real(ffp);
% rhs4 = E2d1d'*M02d'*imag(ffp);
% rhs = [rhs1;rhs2;rhs3;rhs4];
% % Walls
% ind_walls = [];                                                           1171
% ind_out = ind_walls;
% lhs(:,ind_out) = []; lhs(ind_out,:) = []; rhs(ind_out,:) = []; clear tmp;
% display('Constructing Solution Done')
% % Calculate Solution
% sol = lhs\rhs;                                                            1176
% display('Solution Done')
% SOL = zeros((MESH.nvar.n+MESH.nvar.e)*2,1);
% tmp = false((MESH.nvar.n+MESH.nvar.e)*2,1); tmp(ind_out) = true;
% SOL(~tmp,1)=sol;
% sol=SOL;                                                                  1181

%% Divide solution in blocks
phi1 = cell(6,1); ux1=phi1; uy1=ux1; vx1=ux1; vy1=ux1; psi1=ux1;
for block=1:6
    if nDeriv                                                             1186
        phi1{block} = phi( [icn1{block};icn1{block}+MESH.nn_g;icn1{block}+
            MESH.nn_g*2;icn1{block}+MESH.nn_g*3] );
        if eqn>0
            ux1{block} = ux( [icey1{block};icn1{block}+MESH.ne_g_v;icey1{block}+
                MESH.nn_g+MESH.ne_g_v;icn1{block}+MESH.nn_g+MESH.ne_g_v*2] );
            uy1{block} = uy( [icex1{block};icex1{block}+MESH.ne_g_h;icn1{block}+
                MESH.ne_g_h*2;icn1{block}+MESH.nn_g+MESH.ne_g_h*2] );
            vx1{block} = vx( [icex1{block};icex1{block}+MESH.ne_g_h;icn1{block}+  1191
                MESH.ne_g_h*2;icn1{block}+MESH.nn_g+MESH.ne_g_h*2] );
            vy1{block} = vy( [icey1{block};icn1{block}+MESH.ne_g_v;icey1{block}+
                MESH.nn_g+MESH.ne_g_v;icn1{block}+MESH.nn_g+MESH.ne_g_v*2] );
```

```
            psi1{block} = psi( [icv1{block};icey1{block}+MESH.nv_g;icex1{block}+
                MESH.nv_g+MESH.ne_g_v;icn1{block}+MESH.nv_g+MESH.ne_g] );
        end
    else
        phi1{block} = phi( icn1{block} );
        if eqn>0
            ux1{block} = ux( icey1{block} );
            uy1{block} = uy( icex1{block} );
            vx1{block} = vx( icex1{block} );
            vy1{block} = vy( icey1{block} );
            psi1{block} = psi( icv1{block} );
        end
    end
%     if block>1
%         phi1{block} = phi( icn( mesh{block-1}.nn_g_cs+1:mesh{block}.nn_g_cs ) )
    ;
%         ux1{block} = ux(icey((mesh{block-1}.ne_g_v_cs+1:mesh{block}.ne_g_v_cs)
    '));
%         uy1{block} = uy(icex((mesh{block-1}.ne_g_h_cs+1:mesh{block}.ne_g_h_cs)
    '));
%         vx1{block} = vx(icex((mesh{block-1}.ne_g_h_cs+1:mesh{block}.ne_g_h_cs)
    '));
%         vy1{block} = vy(icey((mesh{block-1}.ne_g_v_cs+1:mesh{block}.ne_g_v_cs)
    '));
%         psi1{block} = psi((mesh{block-1}.nv_g_cs+1:mesh{block}.nv_g_cs)');
%     else
%         phi1{block} = phi( icn( 1:mesh{block}.nn_g ) );
%         ux1{block} = ux(icey((1:mesh{block}.ne_g_v)'));
%         uy1{block} = uy(icex((1:mesh{block}.ne_g_h)'));
%         vx1{block} = vx(icex((1:mesh{block}.ne_g_h)'));
%         vy1{block} = vy(icey((1:mesh{block}.ne_g_v)'));
%         psi1{block} = psi((1:mesh{block}.nv_g)');
%     end

end

%% Plot initial solution
if plotje && eqn>0
    ux_l = []; uy_b = [];
    figure(); for tmp = 1:4, h1(tmp) = subplot(2,2,tmp); hold on; end
    for block=plot_bl
        z1 = vec2mat(phi1{block}(1:mesh{block}.nn_g),mesh{block}.N*P+1);
        xe_v = vec2mat(mesh{block}.edgesy(:,3),mesh{block}.N*P+1);
        ye_v = vec2mat(mesh{block}.edgesy(:,4),mesh{block}.N*P+1);
        z2 = vec2mat(ux1{block}(1:mesh{block}.ne_g_v),mesh{block}.N*P+1);
        if sum(block==[1 3 5])
            ux_l = [ux_l;z2(:,1)];
        end
        xe_h = vec2mat(mesh{block}.edgesx(:,3),mesh{block}.N*P);
        ye_h = vec2mat(mesh{block}.edgesx(:,4),mesh{block}.N*P);
        z3 = vec2mat(uy1{block}(1:mesh{block}.ne_g_h),mesh{block}.N*P);
        if sum(block==[1 2])
            uy_b = [uy_b;z3(1,:)'];
        end
        subplot(h1(1)); surf(mesh{block}.XGrid,mesh{block}.YGrid,real(z1),'
            LineStyle','none'); view(0,90); title('\phi_{Re}'); %zlim([-1 1]);
        subplot(h1(2)); surf(mesh{block}.XGrid,mesh{block}.YGrid,real(Fun.PHI(
```

```
            mesh{block}.XGrid,mesh{block}.YGrid)),'LineStyle','none'); view(0,90);
             title('\phi_{Im}'); %zlim([-1 1]);
        subplot(h1(3)); surf(mesh{block}.XGrid,mesh{block}.YGrid,imag(z1),'
            LineStyle','none'); view(0,90); title('\phi_{Re}'); %zlim([-1 1]);
        subplot(h1(4)); surf(mesh{block}.XGrid,mesh{block}.YGrid,imag(Fun.PHI(
            mesh{block}.XGrid,mesh{block}.YGrid)),'LineStyle','none'); view(0,90);
             title('\phi_{Im}'); %zlim([-1 1]);
%         if mesh{block}.M*P>1
%             subplot(h1(1)); surf(xe_v,ye_v,real(z2),'LineStyle','none'); view
    (0,90);  title('u_{x,Re}')
%             subplot(h1(2)); surf(xe_v,ye_v,imag(z2),'LineStyle','none'); view
    (0,90);  title('u_{x,Im}')
%             subplot(h1(3)); surf(xe_h,ye_h,real(z3),'LineStyle','none'); view
    (0,90);  title('u_{y,Re}')
%             subplot(h1(4)); surf(xe_h,ye_h,imag(z3),'LineStyle','none'); view
    (0,90);  title('u_{y,Im}')
%         end
        clear xe_h xe_v ye_h ye_v z1 z2 z3
    end
end

%% Interpolate Fields
phi_i = cell(6,1); ux_i = phi_i; uy_i = ux_i; vx_i = ux_i; vy_i = ux_i; psi_i =
    ux_i;
for block=1:6
    tmp0 = interpolate2D_0form_v1(mesh{block},num_i);
    if nDeriv
        phi_tmp = [vec2mat(phi1{block}(1:end/4),mesh{block}.N*P+1) vec2mat(phi1{
            block}(end/4+1:end/2),mesh{block}.N*P+1);
            vec2mat(phi1{block}(end/2+1:end/4*3),mesh{block}.N*P+1) vec2mat(phi1{
                block}(end/4*3+1:end),mesh{block}.N*P+1)];
    else
        phi_tmp = vec2mat(phi1{block},mesh{block}.N*P+1);
    end
    phi_i{block}.i = tmp0.h_eta_i'*phi_tmp*tmp0.h_xi_i;

    if eqn>0
        tmp1 = interpolate2D_1form_v1(mesh{block},num_i);
        if nDeriv
            ux_tmp00 = vec2mat(ux1{block}(1:mesh{block}.ne_g_v),mesh{block}.N*P
                +1);
            ux_tmp10 = vec2mat(ux1{block}(mesh{block}.ne_g_v+1:mesh{block}.ne_g_v
                *2),mesh{block}.N*P+1);
            ux_tmp01 = vec2mat(ux1{block}(mesh{block}.ne_g_v*2+1:mesh{block}
                .ne_g_v*2+mesh{block}.nn_g),mesh{block}.N*P+1) ;
            ux_tmp11 = vec2mat(ux1{block}(mesh{block}.ne_g_v*2+mesh{block}.nn_g+1
                :end),mesh{block}.N*P+1);
            ux_tmp = [ux_tmp00 ux_tmp10; ux_tmp01 ux_tmp11];
            ux_i{block}.i00 = tmp1.w_eta'*(tmp1.e_eta_f_i'*ux_tmp00*tmp0.h_xi0_i)
                ;
            ux_i{block}.i10 = tmp1.w_eta'*(tmp1.e_eta_f_i'*ux_tmp10*tmp0.hp_xi1_i
                );
            ux_i{block}.i01 = (tmp0.hp_eta1_i'*ux_tmp01*tmp0.h_xi0_i);
            ux_i{block}.i11 = (tmp0.hp_eta1_i'*ux_tmp11*tmp0.hp_xi1_i);
            ux_i{block}.i = ([tmp1.e_eta_i;tmp0.hp_eta1_i]'*ux_tmp*tmp0.h_xi_i);
        else
            ux_tmp = vec2mat(ux1{block},mesh{block}.N*P+1);
```

```
            ux_i{block}.i00 = tmp1.w_eta'*(tmp1.e_eta_f_i'*ux_tmp*tmp0.h_xi_i); %
            ux_i{block}.i = (tmp1.e_eta_i'*ux_tmp*tmp0.h_xi_i);
            uy_tmp = vec2mat(uy1{block},mesh{block}.N*P);
            uy_i{block}.i00 = (tmp0.h_eta_i'*uy_tmp*tmp1.e_xi_f_i)*tmp1.w_xi; %
            uy_i{block}.i = (tmp0.h_eta_i'*uy_tmp*tmp1.e_xi_i);
            vx_tmp = vec2mat(vx1{block},mesh{block}.N*P);
            vx_i{block}.i00 = (tmp0.h_eta_i'*vx_tmp*tmp1.e_xi_f_i)*tmp1.w_xi; %
            vx_i{block}.i = (tmp0.h_eta_i'*vx_tmp*tmp1.e_xi_i);
            vy_tmp = vec2mat(vy1{block},mesh{block}.N*P+1);
            vy_i{block}.i00 = tmp1.w_eta'*(tmp1.e_eta_f_i'*vy_tmp*tmp0.h_xi_i); %
            vy_i{block}.i = (tmp1.e_eta_i'*vy_tmp*tmp0.h_xi_i);
            psi_tmp = vec2mat(psi1{block},mesh{block}.N*P);
            psi_i{block}.i00 = tmp1.w_eta'*(tmp1.e_eta_f_i'*psi_tmp*tmp1.e_xi_f_i
                )*tmp1.w_xi; %
            psi_i{block}.i = (tmp1.e_eta_i'*psi_tmp*tmp1.e_xi_i);
        end
    end
end
clear *tmp*
disp('Interpolation Done')

%% Single line at distance x
if diffr
    [~,ind] = min(abs(mesh{2}.XfGLL-(L+dx(1))));
    err = mesh{2}.XfGLL(ind)-(L+dx(1));
%     display(err);
    Y = 0; PHI = 0;
    for block = 2:2:6
        Y = [Y(1:end-1); mesh{block}.YfGLL'];
        PHI = [PHI(1:end-1); phi_i{block}.i(:,ind)];
    end
    W = domain.dy*yf; % size slit
    x = L; % distance from slit
    a = 1; % amplitude incoming wave
    lambda = 1/wvnr; % wavelength
    z = Y-domain.dy/2;
    PHI_ex = a.*W.*sinc(pi.*W.*x./lambda./z);

    if plotje
        figure(); plot(Y,sqrt(real(PHI).^2+imag(PHI).^2),'b-',Y,sqrt(real(PHI_ex)
            .^2+imag(PHI_ex).^2),'w--','LineWidth',2); grid on;
        xlabel('y'); ylabel('||\phi||');
        title(['Wave magnitude at distance, L=',num2str(L),' from opening']);
    end
    ConvCheck{P}.Y = Y;
    ConvCheck{P}.PHI = PHI;

    % check symmetry
    tmp = max(real(PHI-flipud(PHI)))/max(real(PHI));
    fprintf('Maximum relative symmetricity error: %.2e%%',tmp)
end

%% Plot Interpolated Results
if plotje
figure(); hold on;
for block=plot_bl
%     surf(mesh{block}.XfGrid,mesh{block}.YfGrid,sqrt(real(phi_i{block}.i).^2+
```

```
    imag(phi_i{block}.i).^2),'LineStyle','none'); clear x y z
    surf(mesh{block}.XfGrid,mesh{block}.YfGrid,real(phi_i{block}.i),'LineStyle','
        none'); clear x y z
    hold on;
    for i=1:(diff(domain.ys([1 4]))*yf*wvnr)
        sintheta(i,1) = i/wvnr/diff(domain.ys(2:3));
        a = sqrt(diff(domain.xs(2:3))^2/(1/sintheta(i)^2-1));
%         plot3(domain.xs(2:3)',[mean(domain.ys(2:3)) mean(domain.ys(2:3))+a],[5
    5],'w'); plot3(domain.xs(2:3)',[mean(domain.ys(2:3)) mean(domain.ys(2:3))-a
    ],[5 5],'w');
    end
    plot3([domain.xs(2) domain.xs(2)],domain.ys(1:2)',[1 1],'-k','LineWidth',1);
    plot3([domain.xs(2) domain.xs(2)],domain.ys(3:4)',[1 1],'-k','LineWidth',1);
    xlim(domain.xs([1 3])); ylim(domain.ys([1 4]));
end
colormap jet
end

%% Error
phi_it = []; dif = [];
for block=1:6
    dif = [dif;real(mat2vec(phi_i{block}.i)-Fun.PHI(mesh{block}.Xf,mesh{block}.Yf
        ))];
    phi_it = [phi_it;mat2vec(real(phi_i{block}.i))];
end
err = norm(dif)/norm(phi_it); display(err)
count = count+1;
ERR(count,:) = [MESH.nvar.e*2+MESH.nvar.n*2 P nDeriv err wvnr Nwv];

%% Animation
if anim
scrsz = get(groot,'ScreenSize');
figure('Position',scrsz)
m = 10;
axis tight
ax = gca;
% ax.DataAspectRatio = [max(domain.xs) max(domain.ys) 1];
view(10,80);
axis off
for j = 1:m*5
    for block=plot_bl
        uu = real((phi_i{block}.i)*exp(-j*2*pi/m*sqrt(-1)));
        surf(mesh{block}.XfGrid,mesh{block}.YfGrid,uu,'LineStyle','none'); hold
            on;
    end
%     for block=plot_bl
%         uu = real(exp(-j*2*pi/m*sqrt(-1))*conj(Fun.PHI(mesh{block}.XfGrid,mesh{
    block}.YfGrid)));
%         surf(mesh{block}.XfGrid,mesh{block}.YfGrid,uu,'LineStyle','none'); hold
     on;
%     end
    colormap jet
    hold off;
%     caxis([-2 2]);
    axis tight
    ax = gca;
%     ax.DataAspectRatio = [max(domain.xs) max(domain.ys) 1];
```

```
    view(10,80);
    axis off
    pause(0.1);
end
end

end % P
end % nDeriv

%% Plot Errors
if count>4
    figure();
    loglog(ERR(1:end/2,6),ERR(1:end/2,4),ERR(end/2+1:end,6),ERR(end/2+1:end,4));
    legend('HH=0','HH=1');
    grid on;
end

disp((MESH.nvar.e+MESH.nvar.n)*2)
```

B.2 Hermite_solver2D_square.m

```
clearvars -except other
close all
clc

addpath('../functions/Hermite')
addpath('../functions')

indA = 1;
saveFig = 0;
HH = 1;
A = 0; % directional derivative?
for nDeriv = 1%[0 1]
    plane_wave = 1;
    indB = 1;
    for P = 1%[1 2 3/2 5/2]*(1+~nDeriv)%transp(nonzeros(unique([(2:4)*~nDeriv
        (1:2)*nDeriv])));%(1)*(1+~nDeriv);
        if nDeriv==1 && P==3/2, break; end
        indC = 1;
        for wvnr = 10%[1:4 6:2:20]%2.^(0:3);
            clearvars -except indA saveFig HH A nDeriv plane_wave indB P indC
                wvnr errs errsBAE errs_phi errs_phiBAE errs_ux errs_uxBAE errs_uy
                errs_uyBAE errs_u nVars Ps wvnrs Nwvs nDerivs HHs
            Nwv = 4;
            domain.x = [0 1];
            domain.y = [0 1];

            N = wvnr*Nwv;
            M = wvnr*Nwv;
            omega = (wvnr)*2*pi;
            theta = 1*pi/4;
            gam = (HH-~HH);%*(omega)^2;
            num_i = 4*(P+1)*(nDeriv+1);
%             num_i = ceil((100*2.^(0:0))/(wvnr*Nwv*diff(domain.x)*P+1))*(P+1);
            num_d = 2*(P+1)*(nDeriv+1);
             display([HH nDeriv P wvnr Nwv])
            % Plot basis functions
%             HermitePoly((0:0.01:1)*(diff(domain.x)/N),P,nDeriv,N*2/diff(
    domain.x),1);

            %% Mass Matrices
            [M00,M11,M2d2d,M02d,M11d,M1d1d] = mass_matrices2D_v4_2(P,N,M,nDeriv,
                domain);
            mesh = sizes(N,M,P); mesh.N = N; mesh.M = M; mesh.P = P; mesh.nDeriv
                = nDeriv; mesh.domain = domain;
            [M00_BC.b,M11_BC.b,M01_BC.b] = mass_matrices_2D_v4_2_BCtest(mesh
                ,-1,0);
            [M00_BC.t,M11_BC.t,M01_BC.t] = mass_matrices_2D_v4_2_BCtest(mesh,1,0)
                ;
            [M00_BC.l,M11_BC.l,M01_BC.l] = mass_matrices_2D_v4_2_BCtest(mesh
                ,0,-1);
            [M00_BC.r,M11_BC.r,M01_BC.r] = mass_matrices_2D_v4_2_BCtest(mesh,0,1)
                ;

            %% Element spacing
```

```
% Main elements
mesh.xN = linspace(domain.x(1),domain.x(2),N+1);
mesh.yN = linspace(domain.y(1),domain.y(2),M+1);
% Normal and fine local GLL nodes
mesh.xGLL = ((GLLnodes(P)+1)/2)/N*(domain.x(2)-domain.x(1));
    mesh.XGLL = 0;
mesh.xfGLL = ((GLLnodes(num_i-1)+1)/2)/N*(domain.x(2)-domain.x(1));
    mesh.XfGLL = 0;
mesh.xdGLL = ((GLLnodes(num_d-1)+1)/2)/N*(domain.x(2)-domain.x(1));
    mesh.XdGLL = 0;
% Normal and fine global GLL nodes
for iN=1:N
    mesh.XGLL = [mesh.XGLL(1:end-1) mesh.xGLL+mesh.xN(iN)];
    mesh.XfGLL = [mesh.XfGLL(1:end-1) mesh.xfGLL+mesh.xN(iN)];
    mesh.XdGLL = [mesh.XdGLL(1:end-1) mesh.xdGLL+mesh.xN(iN)];
end
mesh.yGLL = ((GLLnodes(P)+1)/2)/M*(domain.y(2)-domain.y(1));
    mesh.YGLL = 0;
mesh.yfGLL = ((GLLnodes(num_i-1)+1)/2)/M*(domain.y(2)-domain.y(1));
    mesh.YfGLL = 0;
mesh.ydGLL = ((GLLnodes(num_d-1)+1)/2)/M*(domain.y(2)-domain.y(1));
    mesh.YdGLL = 0;
for jM=1:M
    mesh.YGLL = [mesh.YGLL(1:end-1) mesh.yGLL+mesh.yN(jM)];
    mesh.YfGLL = [mesh.YfGLL(1:end-1) mesh.yfGLL+mesh.yN(jM)];
    mesh.YdGLL = [mesh.YdGLL(1:end-1) mesh.ydGLL+mesh.yN(jM)];
end
% Grid nodes
[mesh.XGrid, mesh.YGrid] = meshgrid(mesh.XGLL,mesh.YGLL);
mesh.X = mesh.XGrid'; mesh.X = mesh.X(:);
mesh.Y = mesh.YGrid'; mesh.Y = mesh.Y(:);
[mesh.XfGrid, mesh.YfGrid] = meshgrid(mesh.XfGLL,mesh.YfGLL);
mesh.Xf = mesh.XfGrid'; mesh.Xf = mesh.Xf(:);
mesh.Yf = mesh.YfGrid'; mesh.Yf = mesh.Yf(:);
[mesh.XdGrid, mesh.YdGrid] = meshgrid(mesh.XdGLL,mesh.YdGLL);
mesh.Xd = mesh.XdGrid'; mesh.Xd = mesh.Xd(:);
mesh.Yd = mesh.YdGrid'; mesh.Yd = mesh.Yd(:);
mesh.XY = [mesh.X mesh.Y];
nodes = (1:mesh.nn_g)'; node1 = nodes; node2 = nodes;
node1((1:M*P+1)*(N*P+1)) = [];
node2((0:M*P)*(N*P+1)+1) = [];
mesh.edgesx = [node1 node2 (mesh.X(node1)+mesh.X(node2))/2 mesh.Y(
    node1)];
node3 = (1:(mesh.nn_g-(N*P+1)))'; node4 = ((N*P+2):mesh.nn_g)';
mesh.edgesy = [node3 node4 mesh.X(node3) (mesh.Y(node3)+mesh.Y(node4)
    )/2];
clearvars -regexp node[1234s]
edgesy = (1:mesh.ne_g_v)'; edge3 = edgesy; edge4 = edgesy;
edge1 = (1:(mesh.ne_g_h-N*P))'; edge2 = ((N*P+1):mesh.ne_g_h)';
edge3((1:M*P)*(N*P+1)) = [];
edge4((0:(M*P-1))*(N*P+1)+1) = [];
mesh.volumes = [edge1 edge2 edge3 edge4 mesh.edgesx(edge1,3)
    mesh.edgesy(edge3,4)];
clearvars -regexp edge[1234{sy}]
mesh.nvar.n = mesh.nn_g*(nDeriv+1)^2;
mesh.nvar.e_h = mesh.ne_g_h+nDeriv*(mesh.nn_g*2+mesh.ne_g_h);
mesh.nvar.e_v = mesh.ne_g_v+nDeriv*(mesh.nn_g*2+mesh.ne_g_v);
```

```
mesh.nvar.e = mesh.ne_g+nDeriv*(mesh.nn_g*4+mesh.ne_g);
mesh.nvar.v = mesh.nv_g+nDeriv*(mesh.nn_g+mesh.ne_g);

%% Create Incidence Matrices
E10x1 = sparse(1:mesh.ne_g_h,mesh.edgesx(:,1),-1,mesh.ne_g_h,
    mesh.nn_g)+...
    sparse(1:mesh.ne_g_h,mesh.edgesx(:,2),1,mesh.ne_g_h,mesh.nn_g);
E10y1 = sparse(1:mesh.ne_g_v,mesh.edgesy(:,1),-1,mesh.ne_g_v,
    mesh.nn_g)+...
    sparse(1:mesh.ne_g_v,mesh.edgesy(:,2),1,mesh.ne_g_v,mesh.nn_g);
E21x1 = sparse(1:mesh.nv_g,mesh.volumes(:,1),-1,mesh.nv_g,mesh.ne_g_h
    )+...
    sparse(1:mesh.nv_g,mesh.volumes(:,2),1,mesh.nv_g,mesh.ne_g_h);
E21y1 = sparse(1:mesh.nv_g,mesh.volumes(:,3),1,mesh.nv_g,mesh.ne_g_v)
    +...
    sparse(1:mesh.nv_g,mesh.volumes(:,4),-1,mesh.nv_g,mesh.ne_g_v);
if nDeriv
    E10x = blkdiag(E10x1,sparse(1:mesh.nn_g,1:mesh.nn_g,1),E10x1,
        sparse(1:mesh.nn_g,1:mesh.nn_g,1));
    E10y = blkdiag(E10y1,E10y1,sparse(1:mesh.nn_g,1:mesh.nn_g,1),
        sparse(1:mesh.nn_g,1:mesh.nn_g,1));
    E21x = blkdiag(E21x1,E10y1,sparse(1:mesh.ne_g_h,1:mesh.ne_g_h,1),
        sparse(1:mesh.nn_g,1:mesh.nn_g,1));
    E21y = blkdiag(E21y1,-sparse(1:mesh.ne_g_v,1:mesh.ne_g_v,1),-
        E10x1,-sparse(1:mesh.nn_g,1:mesh.nn_g,1));
    E1d0dx = blkdiag(E10y1,E10y1,sparse(1:mesh.nn_g,1:mesh.nn_g,1),
        sparse(1:mesh.nn_g,1:mesh.nn_g,1)); % sparse((mesh.nn_g+
        mesh.ne_g_v)*2,mesh.nn_g*4,0); %
    E1d0dy = blkdiag(-E10x1,sparse(1:mesh.nn_g,1:mesh.nn_g,-1),-E10x1
        ,sparse(1:mesh.nn_g,1:mesh.nn_g,-1)); % sparse((mesh.nn_g+
        mesh.ne_g_h)*2,mesh.nn_g*4,0); %
    E2d1dx = blkdiag(-E21y1,sparse(1:mesh.ne_g_v,1:mesh.ne_g_v,1),
        E10x1,sparse(1:mesh.nn_g,1:mesh.nn_g,1)); % E21y;Iey;-E10x;In
    E2d1dy = blkdiag(E21x1,E10y1,sparse(1:mesh.ne_g_h,1:mesh.ne_g_h
        ,1),sparse(1:mesh.nn_g,1:mesh.nn_g,1)); % E21x;E10y;Iex;In
else
    E10x = E10x1;
    E10y = E10y1;
    E21x = E21x1;
    E21y = E21y1;
    E1d0dx = E10y1;
    E1d0dy = -E10x1;
    E2d1dx = -E21y1;
    E2d1dy = E21x1;
end
E10 = [E10x;E10y]; E1d0d = [E1d0dx;E1d0dy];
E21 = [E21x E21y]; E2d1d = [E2d1dx E2d1dy];
% E21*E10
% E2d1d*E1d0d
clearvars -regexp E[12][01][\w*] E[12]d[01]d[\w*]
E10x = E10(1:mesh.nvar.e_h,:);
E10y = E10(mesh.nvar.e_h+1:end,:);

%% BC
bc = boundaries(mesh);
n1_l = @(x,y) -(x==min(domain.x)); n1_r = @(x,y) (x==max(domain.x));
    n1 = @(x,y) n1_l(x,y)+n1_r(x,y);
```

```
n2_t = @(x,y) (y==max(domain.y)); n2_b = @(x,y) -(y==min(domain.y));
    n2 = @(x,y) n2_t(x,y)+n2_b(x,y);
[hx, hpx, ex] = HermitePoly((GLLnodes(P)+1)/2*(diff(domain.x)/N),P,
    nDeriv,N*2/diff(domain.x),0);
[hy, hpy, ey] = HermitePoly((GLLnodes(P)+1)/2*(diff(domain.y)/M),P,
    nDeriv,M*2/diff(domain.y),0);
t_l = repmat(ey(:,1)',M*P+1,1);
t_r = repmat(ey(:,end)',M*P+1,1);
t_b = repmat(ex(:,1),1,N*P+1);
t_t = repmat(ex(:,end),1,N*P+1);
if nDeriv
    tmp.Xev = [mesh.edgesy(:,3);mesh.edgesy(:,3);mesh.X;mesh.X];
    tmp.Yev = [mesh.edgesy(:,4);mesh.edgesy(:,4);mesh.Y;mesh.Y];
    tmp.Xeh = [mesh.edgesx(:,3);mesh.X;mesh.edgesx(:,3);mesh.X];
    tmp.Yeh = [mesh.edgesx(:,4);mesh.Y;mesh.edgesx(:,4);mesh.Y];
    mesh.Xev = [tmp.Xev;tmp.Xeh+0.1243];
    mesh.Yev = [tmp.Yev;tmp.Yeh+0.1243];
    mesh.Xeh = [tmp.Xev+0.1243;tmp.Xeh];
    mesh.Yeh = [tmp.Yev+0.1243;tmp.Yeh]; clear tmp;
else
    mesh.Xev = [mesh.edgesy(:,3);mesh.edgesx(:,3)]; mesh.Xeh =
        mesh.Xev;
    mesh.Yev = [mesh.edgesy(:,4);mesh.edgesx(:,4)]; mesh.Yeh =
        mesh.Yev;
end

%% Exact Solution and External forcing
Fun.PHI = @(x,y) exp(1i*omega*(x*cos(theta)+y*sin(theta)));
Fun.PHIpx = @(x,y) 1i*omega*cos(theta)*Fun.PHI(x,y);
Fun.PHIpy = @(x,y) 1i*omega*sin(theta)*Fun.PHI(x,y);
Fun.PHIpxy = @(x,y) -omega^2*cos(theta)*sin(theta)*Fun.PHI(x,y);
Fun.Ux = @(x,y) -Fun.PHI(x,y)*cos(theta);
Fun.Uxpx = @(x,y) -Fun.PHIpx(x,y)*cos(theta);
Fun.Uxpy = @(x,y) -Fun.PHIpy(x,y)*cos(theta);
Fun.Uxpxy = @(x,y) -Fun.PHIpxy(x,y)*cos(theta);
Fun.Uy = @(x,y) -Fun.PHI(x,y)*sin(theta);
Fun.Uypx = @(x,y) -Fun.PHIpx(x,y)*sin(theta);
Fun.Uypy = @(x,y) -Fun.PHIpy(x,y)*sin(theta);
Fun.Uypxy = @(x,y) -Fun.PHIpxy(x,y)*sin(theta);
Fun.Vx = @(x,y) -Fun.PHIpx(x,y);
Fun.Vy = @(x,y) -Fun.PHIpy(x,y);
Fun.PSI = @(x,y) -Fun.Uxpx(x,y)-Fun.Uypy(x,y);
if ~plane_wave
    Fun.PHI = @(x,y) sin(omega*x).*sin(omega*y);
    Fun.PHIpx = @(x,y) omega*cos(omega*x).*sin(omega*y);
    Fun.PHIpy = @(x,y) omega*sin(omega*x).*cos(omega*y);
    Fun.PHIpxy = @(x,y) omega^2*cos(omega*x).*cos(omega*y);
    Fun.Ux = @(x,y) 1i*cos(omega*x).*sin(omega*y);
    Fun.Uxpx = @(x,y) -1i*omega*sin(omega*x).*sin(omega*y);
    Fun.Uxpy = @(x,y) 1i*omega*cos(omega*x).*cos(omega*y);
    Fun.Uxpxy = @(x,y) -1i*omega^2*sin(omega*x).*cos(omega*y);
    Fun.Uy = @(x,y) 1i*sin(omega*x).*cos(omega*y);
    Fun.Uypx = @(x,y) 1i*omega*cos(omega*x).*cos(omega*y);
    Fun.Uypy = @(x,y) -1i*omega*sin(omega*x).*sin(omega*y);
    Fun.Uypxy = @(x,y) -1i*omega^2*cos(omega*x).*sin(omega*y);
end
g = @(x,y) -Fun.PHI(x,y).*(n1(x,y)*cos(theta)+n2(x,y)*sin(theta)-1);
```

```
gx = @(x,y) -Fun.PHI(x,y).*(n1(x,y)*cos(theta)-1);
gy = @(x,y) -Fun.PHI(x,y).*(n2(x,y)*sin(theta)-1);
if nDeriv
    % g
    gpx = @(x,y) -Fun.PHIpx(x,y).*(n1(x,y)*cos(theta)+n2(x,y)*sin(
        theta)-1);
    gpy = @(x,y) -Fun.PHIpy(x,y).*(n1(x,y)*cos(theta)+n2(x,y)*sin(
        theta)-1);
    gpxy = @(x,y) -Fun.PHIpxy(x,y).*(n1(x,y)*cos(theta)+n2(x,y)*sin(
        theta)-1);
    % gx
    gxpx = @(x,y) -Fun.PHIpx(x,y).*(n1(x,y)*cos(theta)-1);
    gxpy = @(x,y) -Fun.PHIpy(x,y).*(n1(x,y)*cos(theta)-1);
    gxpxy = @(x,y) -Fun.PHIpxy(x,y).*(n1(x,y)*cos(theta)-1);
    % gy
    gypx = @(x,y) -Fun.PHIpx(x,y).*(n2(x,y)*sin(theta)-1);
    gypy = @(x,y) -Fun.PHIpy(x,y).*(n2(x,y)*sin(theta)-1);
    gypxy = @(x,y) -Fun.PHIpxy(x,y).*(n2(x,y)*sin(theta)-1);
    % Assemble
    g = @(x,y) [g(x,y);gpx(x,y);gpy(x,y);gpxy(x,y)];
    gx = @(x,y) [gx(x,y);gxpx(x,y);gxpy(x,y);gxpxy(x,y)];
    gy = @(x,y) [gy(x,y);gypx(x,y);gypy(x,y);gypxy(x,y)];
end

if ~plane_wave
    Fun.F = @(x,y) (2*omega^2+gam)/(1i*omega)*Fun.PHI(x,y);
    Fun.Fpx = @(x,y) (2*omega^2+gam)/(1i*omega)*Fun.PHIpx(x,y);
    Fun.Fpy = @(x,y) (2*omega^2+gam)/(1i*omega)*Fun.PHIpy(x,y);
    Fun.Fpxy = @(x,y) (2*omega^2+gam)/(1i*omega)*Fun.PHIpxy(x,y);
elseif plane_wave
    Fun.F = @(x,y) omega^2*(1-gam)/(1i*omega)*Fun.PHI(x,y);
    Fun.Fpx = @(x,y) omega^2*(1-gam)/(1i*omega)*Fun.PHIpx(x,y);
    Fun.Fpy = @(x,y) omega^2*(1-gam)/(1i*omega)*Fun.PHIpy(x,y);
    Fun.Fpxy = @(x,y) omega^2*(1-gam)/(1i*omega)*Fun.PHIpxy(x,y);
end
f = Fun.F(mesh.X,mesh.Y);
if nDeriv>0
    fpx = Fun.Fpx(mesh.X,mesh.Y);
    fpy = Fun.Fpy(mesh.X,mesh.Y);
    fpxy = Fun.Fpxy(mesh.X,mesh.Y);
    ffp = [f;fpx;fpy;fpxy];
else
    ffp=f;
end

% Normals
nx_be = sparse(mesh.nvar.e_v,1,0);
nx_te = sparse(mesh.nvar.e_v,1,0);
ny_le = sparse(mesh.nvar.e_h,1,0);
ny_re = sparse(mesh.nvar.e_h,1,0);
nx_bn = sparse(mesh.nvar.n,1,0);
nx_tn = sparse(mesh.nvar.n,1,0);
ny_ln = sparse(mesh.nvar.n,1,0);
ny_rn = sparse(mesh.nvar.n,1,0);
nx_le = sparse(n1_l(mesh.edgesy(:,3),mesh.edgesy(:,4)));
nx_re = sparse(n1_r(mesh.edgesy(:,3),mesh.edgesy(:,4)));
ny_te = sparse(n2_t(mesh.edgesx(:,3),mesh.edgesx(:,4)));
```

```
ny_be = sparse(n2_b(mesh.edgesx(:,3),mesh.edgesx(:,4)));
nx_ln = sparse(n1_l(mesh.X,mesh.Y));                                    242
nx_rn = sparse(n1_r(mesh.X,mesh.Y));
ny_tn = sparse(n2_t(mesh.X,mesh.Y));
ny_bn = sparse(n2_b(mesh.X,mesh.Y));
if nDeriv
    tmp.nx_le = [nx_le;nx_le;nx_ln;nx_ln];                              247
    tmp.nx_re = [nx_re;nx_re;nx_rn;nx_rn];
    tmp.ny_te = [ny_te;ny_tn;ny_te;ny_tn];
    tmp.ny_be = [ny_be;ny_bn;ny_be;ny_bn];
    tmp.nx_ln = [nx_ln;nx_ln;nx_ln;nx_ln];
    tmp.nx_rn = [nx_rn;nx_rn;nx_rn;nx_rn];                              252
    tmp.ny_tn = [ny_tn;ny_tn;ny_tn;ny_tn];
    tmp.ny_bn = [ny_bn;ny_bn;ny_bn;ny_bn];
    nx_le = tmp.nx_le; nx_re = tmp.nx_re;
    ny_te = tmp.ny_te; ny_be = tmp.ny_be;
    nx_ln = tmp.nx_ln; nx_rn = tmp.nx_rn;                               257
    ny_tn = tmp.ny_tn; ny_bn = tmp.ny_bn; clear tmp;
end

%% System
if plane_wave                                                           262
    if A
        Re_phi_phi = (E10'*M11*E10 + omega^2.*M00 - (...
            +M01_BC.r(:,1:mesh.nvar.e_v)*(-E10y)*sin(theta)/(cos(
                theta))...
            +M01_BC.t(:,mesh.nvar.e_v+1:end)*(-E10x)*cos(theta)/(sin(
                theta))) );
        Re_phi_u = sparse(mesh.nvar.n,mesh.nvar.e,0);                   267
        Im_phi_phi = -omega*(...
            +M00_BC.l...
            +M00_BC.b...
            +M00_BC.r/(cos(theta))...
            +M00_BC.t/(sin(theta)) );                                   272
        Im_phi_u = (gam/omega-omega)*M02d*E2d1d;
        Re_u_u = (E2d1d'*M2d2d*E2d1d+omega^2*M1d1d);
        Re_u_phi = +M11_BC.r(:,1:mesh.nvar.e_v)*(-E10y)*sin(theta)...
            +M11_BC.t(:,mesh.nvar.e_v+1:end)*(-E10x)*cos(theta);
        Im_u_u = omega*(...                                             277
            +M11_BC.l.*([nx_le;ny_le]*[nx_le' ny_le'])...%*(-cos(
                theta))...
            +M11_BC.b.*([nx_be;ny_be]*[nx_be' ny_be'])...%*(-sin(
                theta))...
            +M11_BC.r.*([nx_re;ny_re]*[nx_re' ny_re'])*cos(theta)...
            +M11_BC.t.*blkdiag(nx_te*nx_te',(+ny_te)*(+ny_te)')*sin(
                theta) );
        Im_u_phi = (omega-gam/omega)*E2d1d'*M02d';                      282
    else
        Re_phi_phi = (E10'*M11*E10 + omega^2.*M00);
        Re_phi_u = sparse(mesh.nvar.n,mesh.nvar.e,0);
        Im_phi_phi = -omega*gam*(...
            (M00_BC.l + M00_BC.r)...                                    287
            + (M00_BC.b + M00_BC.t) );
        Im_phi_u = -omega*(1+gam)*M02d*E2d1d;
        Re_u_u = (E2d1d'*M2d2d*E2d1d+omega^2*M1d1d);
        Re_u_phi = sparse(mesh.nvar.e,mesh.nvar.n,0);
        Im_u_u = omega*gam*(...                                         292
```

```
                M11_BC.l.*([nx_le;ny_le]*[nx_le' ny_le']) ...
                +M11_BC.r.*([nx_re;ny_re]*[nx_re' ny_re']) ...
                +M11_BC.b.*([nx_be;ny_be]*[nx_be' ny_be']) ...
                +M11_BC.t.*([nx_te;ny_te]*[nx_te' ny_te']));
            Im_u_phi = omega*(1+gam)*E2d1d'*M02d';                          297
        end
        lhs1 = [Re_phi_phi -Im_phi_phi Re_phi_u -Im_phi_u]; % HH: 1
            HelmHoltz 0 Reaction-Diffusion
        lhs2 = [Im_phi_phi Re_phi_phi Im_phi_u Re_phi_u];
        lhs3 = [Re_u_phi -Im_u_phi Re_u_u -Im_u_u];
        lhs4 = [Im_u_phi Re_u_phi Im_u_u Re_u_u];                           302
        lhs = [lhs1;lhs2;lhs3;lhs4]; clearvars -regexp lhs[?] Re_* Im_*
        % Construct rhs
        if A
            Im_phi_rhs = gam/omega*M00*ffp - omega*( ...
                +M00_BC.b * gy(mesh.X,mesh.Y)...%/(-sin(theta))...      307
                +M00_BC.l * gx(mesh.X,mesh.Y) );%/(-cos(theta)) );
            Re_u_rhs = E2d1d'*M02d'*ffp;
            Im_u_rhs = omega*( ...
                M01_BC.b' * gy(mesh.X,mesh.Y) .* [nx_be;ny_be] ...
                +M01_BC.l' * gx(mesh.X,mesh.Y) .*[nx_le;ny_le] );       312
        else
            Im_phi_rhs = gam/omega*M00*ffp - omega*gam*( ...
                (M00_BC.b+M00_BC.t) * gy(mesh.X,mesh.Y) + ...
                (M00_BC.l+M00_BC.r) * gx(mesh.X,mesh.Y) );
            Re_u_rhs = E2d1d'*M02d'*ffp;                                    317
            Im_u_rhs = omega*gam*( ...
                ( M01_BC.b' * gy(mesh.X,mesh.Y) .* [nx_be;ny_be]...
                +M01_BC.t' * gy(mesh.X,mesh.Y) .* [nx_te;ny_te] ) ...
                +( M01_BC.l' * gx(mesh.X,mesh.Y) .* [nx_le;ny_le]...
                +M01_BC.r' * gx(mesh.X,mesh.Y) .* [nx_re;ny_re] ) );    322
        end
        rhs1 = -imag(Im_phi_rhs);
        rhs2 = real(Im_phi_rhs);
        rhs3 = real(Re_u_rhs)-imag(Im_u_rhs);
        rhs4 = imag(Re_u_rhs)+real(Im_u_rhs);                           327
        rhs = [rhs1;rhs2;rhs3;rhs4]; clearvars -regexp rhs[?] Re_* Im_*
        % Walls
        ind_walls = [];
        ind_out = ind_walls;
        lhs(:,ind_out) = []; lhs(ind_out,:) = []; rhs(ind_out,:) = [];  332
            clear tmp;

        % Calculate Solution
        sol = lhs\rhs;
        SOL = zeros(size(lhs,2),1);
        tmp = false(size(lhs,2),1); tmp(ind_out) = true;                337
        SOL(~tmp,1)=sol;
        sol=SOL;
    else
        % Construct lhs [real(phi)|imag(phi)|real(u)|imag(u)]
        lhs1 = [(E10'*M11*E10+omega^2.*M00) sparse(mesh.nvar.n,          342
            mesh.nvar.n,0) sparse(mesh.nvar.n,mesh.nvar.e,0) (omega-gam/
            omega)*M02d*E2d1d]; % HH: 1 HelmHoltz 0 Reaction-Diffusion
        lhs2 = [sparse(mesh.nvar.n,mesh.nvar.n,0) (E10'*M11*E10+omega^2.*
            M00) -(omega-gam/omega)*M02d*E2d1d sparse(mesh.nvar.n,
            mesh.nvar.e,0)];
```

```
lhs3 = [sparse(mesh.nvar.e,mesh.nvar.n,0) -(omega-gam/omega)*
    E2d1d'*M02d' (E2d1d'*M2d2d*E2d1d+omega^2*M1d1d) sparse(
    mesh.nvar.e,mesh.nvar.e,0)];
lhs4 = [(omega-gam/omega)*E2d1d'*M02d' sparse(mesh.nvar.e,
    mesh.nvar.n,0) sparse(mesh.nvar.e,mesh.nvar.e,0) (E2d1d'*M2d2d
    *E2d1d+omega^2*M1d1d)];
lhs = [lhs1;lhs2;lhs3;lhs4];
% Construct rhs                                                    347
rhs1 = -gam/omega*M00*imag(ffp);
rhs2 = gam/omega*M00*real(ffp);
rhs3 = E2d1d'*M02d'*real(ffp);
rhs4 = E2d1d'*M02d'*imag(ffp);
rhs = [rhs1;rhs2;rhs3;rhs4];                                       352
% Walls
tmp_n = find((bc.l_n+bc.r_n+bc.t_n+bc.b_n)>0);
if nDeriv
    ind_wallsn = [[tmp_n;tmp_n+mesh.nn_g;tmp_n+mesh.nn_g*2;tmp_n+
        mesh.nn_g*3];[tmp_n;tmp_n+mesh.nn_g;tmp_n+mesh.nn_g*2;
        tmp_n+mesh.nn_g*3]+mesh.nn_g];
    rhs = rhs-lhs(:,tmp_n+mesh.nn_g)*real(Fun.PHIpx(mesh.X(tmp_n)  357
        ,mesh.Y(tmp_n)));
    rhs = rhs-lhs(:,tmp_n+mesh.nn_g*2)*real(Fun.PHIpy(mesh.X(
        tmp_n),mesh.Y(tmp_n)));
    rhs = rhs-lhs(:,tmp_n+mesh.nn_g*3)*real(Fun.PHIpxy(mesh.X(
        tmp_n),mesh.Y(tmp_n)));
    rhs = rhs-lhs(:,tmp_n+mesh.nn_g+mesh.nn_g)*imag(Fun.PHIpx(
        mesh.X(tmp_n),mesh.Y(tmp_n)));
    rhs = rhs-lhs(:,tmp_n+mesh.nn_g*2+mesh.nn_g)*imag(Fun.PHIpy(
        mesh.X(tmp_n),mesh.Y(tmp_n)));
    rhs = rhs-lhs(:,tmp_n+mesh.nn_g*3+mesh.nn_g)*imag(Fun.PHIpxy(  362
        mesh.X(tmp_n),mesh.Y(tmp_n)));
else
    ind_wallsn = [tmp_n;tmp_n+mesh.nvar.n];
end
ind_out = ind_wallsn;
lhs(:,ind_out) = []; lhs(ind_out,:) = []; rhs(ind_out,:) = [];     367
display('Constructing Solution Done')
% Calculate Solution
sol = lhs\rhs;
display('Solution Done')
SOL = zeros((mesh.nvar.n+mesh.nvar.e)*2,1);                        372
tmp = false((mesh.nvar.n+mesh.nvar.e)*2,1); tmp(ind_out) = true;
SOL(~tmp,1)=sol;
if nDeriv
    SOL(tmp_n+mesh.nn_g,1) = real(Fun.PHIpx(mesh.X(tmp_n),mesh.Y(
        tmp_n)));
    SOL(tmp_n+mesh.nn_g+mesh.nvar.n,1) = imag(Fun.PHIpx(mesh.X(    377
        tmp_n),mesh.Y(tmp_n)));
    SOL(tmp_n+mesh.nn_g*2,1) = real(Fun.PHIpy(mesh.X(tmp_n),
        mesh.Y(tmp_n)));
    SOL(tmp_n+mesh.nn_g*2+mesh.nvar.n,1) = imag(Fun.PHIpy(mesh.X(
        tmp_n),mesh.Y(tmp_n)));
    SOL(tmp_n+mesh.nn_g*3,1) = real(Fun.PHIpxy(mesh.X(tmp_n),
        mesh.Y(tmp_n)));
    SOL(tmp_n+mesh.nn_g*3+mesh.nvar.n,1) = imag(Fun.PHIpxy(mesh.X
        (tmp_n),mesh.Y(tmp_n)));
end                                                                382
```

```
        sol=SOL;
    end
    % Distribute solution over variables
    SOL = [complex(sol(1:mesh.nvar.n),sol(mesh.nvar.n+1:mesh.nvar.n*2));
        complex(sol((2*mesh.nvar.n+1):(2*mesh.nvar.n+mesh.nvar.e)),sol((2*
        mesh.nvar.n+mesh.nvar.e+1):(2*mesh.nvar.n+2*mesh.nvar.e)))];
    % Collect Results                                                    387
    phi = (SOL(1:mesh.nvar.n)); %phi_im = imag(SOL(1:nn_g*(nDeriv+1)^2));
    u = (SOL(mesh.nvar.n+1:end)); ux = u(1:(mesh.ne_g_v+nDeriv*(
        mesh.ne_g_v+mesh.nn_g*2))); uy = u((mesh.ne_g_v+nDeriv*(
        mesh.ne_g_v+mesh.nn_g*2)+1):end);
    vx = -E10x*phi; vy = -E10y*phi; v = [vx;vy]; psi = -E2d1d*u;
    clearvars sol SOL lhs rhs tmp
                                                                       392
    %% BAE
    [M00_BAE, M1d1d_BAE] = mass_matrices2D_BAE_v3(P,N,M,num_d-1,N,M,
        nDeriv,domain);
    lhs1 = blkdiag(M00,M00);
    lhs2 = blkdiag(M1d1d,M1d1d);
    rhs1 = [M00_BAE*real(Fun.PHI(mesh.Xd,mesh.Yd));                    397
            M00_BAE*imag(Fun.PHI(mesh.Xd,mesh.Yd))];
    rhs2 = [M1d1d_BAE*real([Fun.Ux(mesh.Xd,mesh.Yd);Fun.Uy(mesh.Xd,
        mesh.Yd)]);
            M1d1d_BAE*imag([Fun.Ux(mesh.Xd,mesh.Yd);Fun.Uy(mesh.Xd,
                mesh.Yd)])];
    sol1 = lhs1\rhs1;
    sol2 = lhs2\rhs2;                                                  402
    phiBAE = complex(sol1(1:end/2),sol1(end/2+1:end));
    uxBAE = complex(sol2(1:mesh.nvar.e_v),sol2((mesh.nvar.e+1):(
        mesh.nvar.e+mesh.nvar.e_v)));
    uyBAE = complex(sol2((mesh.nvar.e_v+1):mesh.nvar.e),sol2((mesh.nvar.e
        +mesh.nvar.e_v+1):end));

    %% Interpolate                                                     407
%       ux_i = cell(6,1); uy_i = ux_i; vx_i = ux_i; vy_i = ux_i; psi_i =
    ux_i;
    tmp0 = interpolate2D_0form_v1(mesh,num_i);
    if nDeriv
        phi_tmp = [vec2mat(phi(1:end/4),N*P+1) vec2mat(phi(end/4+1:end/2)
            ,N*P+1);
            vec2mat(phi(end/2+1:end/4*3),N*P+1) vec2mat(phi(end/4*3+1:end  412
                ),N*P+1)];
        phiBAE_tmp = [vec2mat(phiBAE(1:end/4),N*P+1) vec2mat(phiBAE(end
            /4+1:end/2),N*P+1);
            vec2mat(phiBAE(end/2+1:end/4*3),N*P+1) vec2mat(phiBAE(end
                /4*3+1:end),N*P+1)];
    else
        phi_tmp = vec2mat(phi,N*P+1);
        phiBAE_tmp = vec2mat(phiBAE,N*P+1);                            417
    end
    phi_i = tmp0.h_eta_i'*phi_tmp*tmp0.h_xi_i; clear phi_tmp;
    phiBAE_i = tmp0.h_eta_i'*phiBAE_tmp*tmp0.h_xi_i; clear phiBAE_tmp;

    tmp1 = interpolate2D_1form_v1(mesh,num_i);                         422
    if nDeriv
        ux_tmp00 = vec2mat(ux(1:mesh.ne_g_v),N*P+1);
        ux_tmp10 = vec2mat(ux(mesh.ne_g_v+1:mesh.ne_g_v*2),N*P+1);
```

```
                ux_tmp01 = vec2mat(ux(mesh.ne_g_v*2+1:mesh.ne_g_v*2+mesh.nn_g),N*
                    P+1) ;
                ux_tmp11 = vec2mat(ux(mesh.ne_g_v*2+mesh.nn_g+1:end),N*P+1);
                ux_tmp = [ux_tmp00 ux_tmp10; ux_tmp01 ux_tmp11];
                ux_i.i00 = tmp1.w_eta'*(tmp1.e_eta_f_i'*ux_tmp00*tmp0.h_xi0_i);
                ux_i.i10 = tmp1.w_eta'*(tmp1.e_eta_f_i'*ux_tmp10*tmp0.hp_xi1_i);
                ux_i.i01 = (tmp0.hp_eta1_i'*ux_tmp01*tmp0.h_xi0_i);
                ux_i.i11 = (tmp0.hp_eta1_i'*ux_tmp11*tmp0.hp_xi1_i);
                ux_i.i = ([tmp1.e_eta_i;tmp0.hp_eta1_i]'*ux_tmp*tmp0.h_xi_i);
                uy_tmp00 = vec2mat(uy(1:mesh.ne_g_h),N*P);
                uy_tmp10 = vec2mat(uy(mesh.ne_g_h+1:(mesh.ne_g_h+mesh.nn_g)),N*P
                    +1);
                uy_tmp01 = vec2mat(uy((mesh.ne_g_h+mesh.nn_g)+1:(mesh.ne_g_h*2+
                    mesh.nn_g)),N*P) ;
                uy_tmp11 = vec2mat(uy((mesh.ne_g_h*2+mesh.nn_g)+1:end),N*P+1);
                uy_tmp = [uy_tmp00 uy_tmp10; uy_tmp01 uy_tmp11];
                uy_i.i = (tmp0.h_xi_i'*uy_tmp*[tmp1.e_eta_i;tmp0.hp_eta1_i]);
                uxBAE_tmp00 = vec2mat(uxBAE(1:mesh.ne_g_v),N*P+1);
                uxBAE_tmp10 = vec2mat(uxBAE(mesh.ne_g_v+1:mesh.ne_g_v*2),N*P+1);
                uxBAE_tmp01 = vec2mat(uxBAE(mesh.ne_g_v*2+1:mesh.ne_g_v*2+
                    mesh.nn_g),N*P+1) ;
                uxBAE_tmp11 = vec2mat(uxBAE(mesh.ne_g_v*2+mesh.nn_g+1:end),N*P+1)
                    ;
                uxBAE_tmp = [uxBAE_tmp00 uxBAE_tmp10; uxBAE_tmp01 uxBAE_tmp11];
                uxBAE_i.i00 = tmp1.w_eta'*(tmp1.e_eta_f_i'*uxBAE_tmp00*
                    tmp0.h_xi0_i);
                uxBAE_i.i10 = tmp1.w_eta'*(tmp1.e_eta_f_i'*uxBAE_tmp10*
                    tmp0.hp_xi1_i);
                uxBAE_i.i01 = (tmp0.hp_eta1_i'*uxBAE_tmp01*tmp0.h_xi0_i);
                uxBAE_i.i11 = (tmp0.hp_eta1_i'*uxBAE_tmp11*tmp0.hp_xi1_i);
                uxBAE_i.i = ([tmp1.e_eta_i;tmp0.hp_eta1_i]'*uxBAE_tmp*tmp0.h_xi_i
                    );
                uyBAE_tmp00 = vec2mat(uyBAE(1:mesh.ne_g_h),N*P);
                uyBAE_tmp10 = vec2mat(uyBAE(mesh.ne_g_h+1:(mesh.ne_g_h+mesh.nn_g)
                    ),N*P+1);
                uyBAE_tmp01 = vec2mat(uyBAE((mesh.ne_g_h+mesh.nn_g)+1:(
                    mesh.ne_g_h*2+mesh.nn_g)),N*P) ;
                uyBAE_tmp11 = vec2mat(uyBAE((mesh.ne_g_h*2+mesh.nn_g)+1:end),N*P
                    +1);
                uyBAE_tmp = [uyBAE_tmp00 uyBAE_tmp10; uyBAE_tmp01 uyBAE_tmp11];
                uyBAE_i.i = (tmp0.h_xi_i'*uyBAE_tmp*[tmp1.e_eta_i;tmp0.hp_eta1_i
                    ]);
%                 vx_tmp00 = vec2mat(vx(1:mesh.ne_g_h),N*P);
%                 vx_tmp10 = vec2mat(vx(mesh.ne_g_h+1:(mesh.ne_g_h+mesh.nn_g)),N*
    P+1);
%                 vx_tmp01 = vec2mat(vx((mesh.ne_g_h+mesh.nn_g)+1:(mesh.ne_g_h*2+
    mesh.nn_g)),N*P) ;
%                 vx_tmp11 = vec2mat(vx((mesh.ne_g_h*2+mesh.nn_g)+1:end),N*P+1);
%                 vx_tmp = [vx_tmp00 vx_tmp10; vx_tmp01 vx_tmp11];
%                 vx_i.i = (tmp0.h_xi_i'*vx_tmp*[tmp1.e_eta_i;tmp0.hp_eta1_i]);
%                 vy_tmp00 = vec2mat(vy(1:mesh.ne_g_v),N*P+1);
%                 vy_tmp10 = vec2mat(vy(mesh.ne_g_v+1:mesh.ne_g_v*2),N*P+1);
%                 vy_tmp01 = vec2mat(vy(mesh.ne_g_v*2+1:mesh.ne_g_v*2+mesh.nn_g),
    N*P+1) ;
%                 vy_tmp11 = vec2mat(vy(mesh.ne_g_v*2+mesh.nn_g+1:end),N*P+1);
%                 vy_tmp = [vy_tmp00 vy_tmp10; vy_tmp01 vy_tmp11];
%                 vy_i.i = ([tmp1.e_eta_i;tmp0.hp_eta1_i]'*vy_tmp*tmp0.h_xi_i);
```

```
%                 psi_tmp00 = vec2mat(psi(1:mesh.nv_g),N*P);
%                 psi_tmp10 = vec2mat(psi(mesh.nv_g+1:(mesh.ne_g_v+mesh.nv_g)),N*
    P+1);
%                 psi_tmp01 = vec2mat(psi((mesh.ne_g_v+mesh.nv_g)+1:(mesh.ne_g+
    mesh.nv_g)),N*P) ;
%                 psi_tmp11 = vec2mat(psi((mesh.ne_g+mesh.nv_g)+1:end),N*P+1);
%                 psi_tmp = [psi_tmp00 psi_tmp10; psi_tmp01 psi_tmp11];
%                 psi_i.i = ([tmp1.e_eta_i;tmp0.hp_eta1_i]'*psi_tmp*[tmp1.e_eta_i
    ;tmp0.hp_eta1_i]);
            else
                ux_tmp = vec2mat(ux,N*P+1);
                ux_i.i00 = tmp1.w_eta'*(tmp1.e_eta_f_i'*ux_tmp*tmp0.h_xi_i); %
                ux_i.i = (tmp1.e_eta_i'*ux_tmp*tmp0.h_xi_i);
                uy_tmp = vec2mat(uy,N*P);
                uy_i.i00 = (tmp0.h_eta_i'*uy_tmp*tmp1.e_xi_f_i)*tmp1.w_xi; %
                uy_i.i = (tmp0.h_eta_i'*uy_tmp*tmp1.e_xi_i);
                uxBAE_tmp = vec2mat(uxBAE,N*P+1);
                uxBAE_i.i00 = tmp1.w_eta'*(tmp1.e_eta_f_i'*uxBAE_tmp*tmp0.h_xi_i)
                    ; %
                uxBAE_i.i = (tmp1.e_eta_i'*uxBAE_tmp*tmp0.h_xi_i);
                uyBAE_tmp = vec2mat(uyBAE,N*P);
                uyBAE_i.i00 = (tmp0.h_eta_i'*uyBAE_tmp*tmp1.e_xi_f_i)*tmp1.w_xi;
                    %
                uyBAE_i.i = (tmp0.h_eta_i'*uyBAE_tmp*tmp1.e_xi_i);
%                 vx_tmp = vec2mat(vx,N*P);
%                 vx_i.i00 = (tmp0.h_eta_i'*vx_tmp*tmp1.e_xi_f_i)*tmp1.w_xi; %
%                 vx_i.i = (tmp0.h_eta_i'*vx_tmp*tmp1.e_xi_i);
%                 vy_tmp = vec2mat(vy,N*P+1);
%                 vy_i.i00 = tmp1.w_eta'*(tmp1.e_eta_f_i'*vy_tmp*tmp0.h_xi_i); %
%                 vy_i.i = (tmp1.e_eta_i'*vy_tmp*tmp0.h_xi_i);
%                 psi_tmp = vec2mat(psi,N*P);
%                 psi_i.i00 = tmp1.w_eta'*(tmp1.e_eta_f_i'*psi_tmp*tmp1.e_xi_f_i)
    *tmp1.w_xi; %
%                 psi_i.i = (tmp1.e_eta_i'*psi_tmp*tmp1.e_xi_i);
            end
            clearvars tmp0 tmp1 ux_tmp* uy_tmp* vx_tmp* vy_tmp* psi_tmp*

            %% Plot mesh points
            if 1
                data = mesh.XY;
                figure(); scatter(data(:,1),data(:,2),'.'); hold on;
                c = cellstr(num2str((1:size(data,1))'));
                text(data(:,1)+(((data(:,1)-0.501)>0)*2-1.75)*0.02,data(:,2)+0.01
                    ,c);
                c = cellstr(num2str((1:size(mesh.edgesx,1))'));
                text(mesh.edgesx(:,3),mesh.edgesx(:,4),c);
                c = cellstr(num2str((1:size(mesh.edgesy,1))'));
                text(mesh.edgesy(:,3),mesh.edgesy(:,4),c);
                clear data;
            end

            %% Plot Matrix
            if 1
                mat = M11d;
                figure; spy(mat);
            end
```

```
%% Plot initial solution
if 1
    ux_l = []; uy_b = [];
    figure(); for tmp = 1:4, h(tmp) = subplot(2,2,tmp); hold on; end
    z1 = vec2mat(phi(1:mesh.nn_g),N*P+1);                                    522
    xe_v = vec2mat(mesh.edgesy(:,3),N*P+1);
    ye_v = vec2mat(mesh.edgesy(:,4),N*P+1);
    z2 = vec2mat(ux(1:mesh.ne_g_v),N*P+1);
    xe_h = vec2mat(mesh.edgesx(:,3),N*P);
    ye_h = vec2mat(mesh.edgesx(:,4),N*P);                                    527
    z3 = vec2mat(uy(1:mesh.ne_g_h),N*P);
    subplot(h(1)); surf(mesh.XGrid,mesh.YGrid,real(z1)); view(0,90);
        title('\phi_{Re}'); %zlim([-1 1]);
    subplot(h(2)); surf(mesh.XGrid,mesh.YGrid,real(Fun.PHI(mesh.XGrid
        ,mesh.YGrid))); view(0,90); title('\phi_{Im}'); %zlim([-1 1]);
    subplot(h(3)); surf(mesh.XGrid,mesh.YGrid,imag(z1)); view(0,90);
        title('\phi_{Re}'); %zlim([-1 1]);
    subplot(h(4)); surf(mesh.XGrid,mesh.YGrid,imag(Fun.PHI(mesh.XGrid   532
        ,mesh.YGrid))); view(0,90); title('\phi_{Im}'); %zlim([-1 1]);
    clear xe_h xe_v ye_h ye_v z1 z2 z3
end

%% Plot interpolatd solution
if 1                                                                         537
    figure; surf(mesh.XfGrid,mesh.YfGrid,real(phi_i),'LineStyle','
        none'); view(0,90); clear x y z; colormap jet
    figure; for tmp = 1:6, h(tmp) = subplot(2,3,tmp); hold on;
        colormap jet; end
    subplot(h(1)); surf(mesh.XfGrid,mesh.YfGrid,real(phi_i),'
        LineStyle','none'); title('\phi_{Re}'); view(15,65); zlim([-1
        1]); grid on;
    subplot(h(2)); surf(mesh.XfGrid,mesh.YfGrid,real(Fun.PHI(
        mesh.XfGrid,mesh.YfGrid)),'LineStyle','none'); title('\phi_{Re
        } exact'); view(15,65); zlim([-1 1]); grid on;
    subplot(h(3)); surf(mesh.XfGrid,mesh.YfGrid,real(phi_i)-real(       542
        Fun.PHI(mesh.XfGrid,mesh.YfGrid)),'LineStyle','none'); title('
        \phi_{Re} difference'); view(15,65); grid on;
    subplot(h(4)); surf(mesh.XfGrid,mesh.YfGrid,imag(phi_i),'
        LineStyle','none'); title('\phi_{Im}'); view(15,65); zlim([-1
        1]); grid on;
    subplot(h(5)); surf(mesh.XfGrid,mesh.YfGrid,imag(Fun.PHI(
        mesh.XfGrid,mesh.YfGrid)),'LineStyle','none'); title('\phi_{Im
        } exact'); view(15,65); zlim([-1 1]); grid on;
    subplot(h(6)); surf(mesh.XfGrid,mesh.YfGrid,imag(phi_i)-imag(
        Fun.PHI(mesh.XfGrid,mesh.YfGrid)),'LineStyle','none'); title('
        \phi_{Im} difference'); view(15,65); grid on;
    clear h
    figure;                                                                  547
    subplot(2,3,1); surf(mesh.XfGrid,mesh.YfGrid,real(ux_i.i),'
        LineStyle','none'); title('Ux'); view(0,90); clear x y z;
        colormap jet
    subplot(2,3,2); surf(mesh.XfGrid,mesh.YfGrid,real(Fun.Ux(
        mesh.XfGrid,mesh.YfGrid)),'LineStyle','none'); title('Ux_{ex}'
        ); view(0,90); clear x y z; colormap jet
    subplot(2,3,3); surf(mesh.XfGrid,mesh.YfGrid,real(uxBAE_i.i),'
        LineStyle','none'); title('Ux_{BAE}'); view(0,90); clear x y z
        ; colormap jet
```

```
            subplot(2,3,4); surf(mesh.XfGrid,mesh.YfGrid,real(uy_i.i),'
                LineStyle','none'); title('Uy'); view(0,90); clear x y z;
                colormap jet
            subplot(2,3,5); surf(mesh.XfGrid,mesh.YfGrid,real(Fun.Uy(
                mesh.XfGrid,mesh.YfGrid)),'LineStyle','none'); title('Uy_{ex}'
                ); view(0,90); clear x y z; colormap jet
            subplot(2,3,6); surf(mesh.XfGrid,mesh.YfGrid,real(uyBAE_i.i),'
                LineStyle','none'); title('Uy_{BAE}'); view(0,90); clear x y z
                ; colormap jet
        end

        %% Error
        phi_v = (mat2vec(phi_i));
        phiBAE_v = (mat2vec(phiBAE_i));
        phi_ex_v = (Fun.PHI(mesh.Xf,mesh.Yf));
        err_phi = norm(phi_v-phi_ex_v)/norm(phi_ex_v); disp('Phi: '); disp(
            err_phi)
        err_phiBAE = norm(phiBAE_v-phi_ex_v)/norm(phi_ex_v); disp('PhiBAE: ')
            ; disp(err_phiBAE)

        ux_v = (mat2vec(ux_i.i));
        uxBAE_v = (mat2vec(uxBAE_i.i));
        ux_ex_v = (Fun.Ux(mesh.Xf,mesh.Yf));
        err_ux = norm(ux_v-ux_ex_v)/norm(ux_ex_v); disp('Ux: '); disp(err_ux)
            ;
        err_uxBAE = norm(uxBAE_v-ux_ex_v)/norm(ux_ex_v); disp('UxBAE: ');
            disp(err_uxBAE);

        uy_v = (mat2vec(uy_i.i));
        uyBAE_v = (mat2vec(uyBAE_i.i));
        uy_ex_v = (Fun.Uy(mesh.Xf,mesh.Yf));
        err_uy = norm(uy_v-uy_ex_v)/norm(uy_ex_v); disp('Uy: '); disp(err_uy)
            ;
        err_uyBAE = norm(uyBAE_v-uy_ex_v)/norm(uy_ex_v); disp('UyBAE: ');
            disp(err_uyBAE);

        a1 = (phi_v-phi_ex_v); a2 = (phiBAE_v-phi_ex_v);
        b1 = (ux_v-ux_ex_v); b2 = (uxBAE_v-ux_ex_v);
        c1 = (uy_v-uy_ex_v); c2 = (uyBAE_v-uy_ex_v);
        d1 = [b1;c1]; d2 = [b2;c2];
        err = sqrt((a1'*a1)+(b1'*b1)+(c1'*c1));
        errBAE = sqrt((a2'*a2)+(b2'*b2)+(c2'*c2));
%         sqrt(abc1/abc2)
%         sqrt((a1'*a1)/(a2'*a2))
%         sqrt(((b1'*b1)+(c1'*c1))/((b2'*b2)+(c2'*c2)))

%         vx_v = real(mat2vec(vx_i.i));
%         vx_ex_v = real(Fun.Vx(mesh.Xf,mesh.Yf));
%         err_vx = norm(vx_v-vx_ex_v)/norm(vx_ex_v); %disp('Vx: '); disp(
    err_vx);
%         vy_v = real(mat2vec(vy_i.i));
%         vy_ex_v = real(Fun.Vy(mesh.Xf,mesh.Yf));
%         err_vy = norm(vy_v-vy_ex_v)/norm(vy_ex_v); %disp('Vy: '); disp(
    err_vy);
%         psi_v = real(mat2vec(psi_i.i));
%         psi_ex_v = real(Fun.PSI(mesh.Xf,mesh.Yf));
%         err_psi = norm(psi_v-psi_ex_v)/norm(psi_ex_v); %disp('Psi: '); disp
```

```
    (err_psi);

            errs(indC,indB,indA) = err;
            errsBAE(indC,indB,indA) = errBAE;
            errs_phi(indC,indB,indA) = err_phi;
            errs_phiBAE(indC,indB,indA) = err_phiBAE;
            errs_ux(indC,indB,indA) = err_ux;
            errs_uxBAE(indC,indB,indA) = err_uxBAE;
            errs_uy(indC,indB,indA) = err_uy;
            errs_uyBAE(indC,indB,indA) = err_uyBAE;
            errs_u(indC,indB,indA) = sqrt(err_ux^2+err_uy^2);
%             errs_vx(indC,indB,indA) = err_vx;
%             errs_vy(indC,indB,indA) = err_vy;
%             errs_v(indC,indB,indA) = sqrt(err_vx^2+err_vy^2);
%             errs_psi(indC,indB,indA) = err_psi;
            nVars(indC,indB,indA) = (mesh.nvar.n+mesh.nvar.e)*2; Ps(indC,indB,
                indA) = P; wvnrs(indC,indB,indA) = omega;
            Nwvs(indC,indB,indA) = Nwv; nDerivs(indC,indB,indA) = nDeriv; HHs(
                indC,indB,indA) = HH;

            disp('Ratio: '); disp(err_phi/err_phiBAE)

            indC=indC+1;
        end
        indB=indB+1;
    end
    indA = indA+1;
end

%% Error plot
if ((indA-1)*(indB-1)*(indC-1)>2 && 1)
    figure;
    if ndims(errs_phi)==3
        for i=1:(indA-1)
            subplot(2,2,1); loglog(wvnrs(:,:,i),errs_phi(:,:,i),'--x'); title('\
                phi'); hold on; grid on; ylabel('L^2 DE');
            subplot(2,2,2); loglog(wvnrs(:,:,i),errs_ux(:,:,i),'--x'); title('ux'
                ); hold on; grid on;
            subplot(2,2,3); semilogx(wvnrs(:,:,i),errs(:,:,i)./errsBAE(:,:,i),'--
                x'); title('Ratio'); hold on; grid on; ylim([0 15]); ylabel('DE/
                BAE'); xlabel('\omega');
            subplot(2,2,4); semilogx(wvnrs(:,:,i),errs_phi(:,:,i)./errs_phiBAE
                (:,:,i),'--x'); title('Ratio_\phi'); hold on; grid on; ylim([0
                15]); xlabel('\omega');
            for nP = 1:4
                subplot(2,2,nP);
                xlim(10.^[floor(log10(min(min(min(wvnrs))))) ceil(log10(max(max(
                    max(wvnrs)))))]);
            end
        end
        hold off;
    else
        subplot(2,2,1); loglog(wvnrs,errs_phi); title('\phi'); hold on; grid on;
        subplot(2,2,2); loglog(wvnrs,errs_ux); title('ux'); hold on; grid on;
        subplot(2,2,3); semilogx(wvnrs,errs./errsBAE); title('Ratio'); hold on;
            grid minor; ylim([0 15]); xlim([5 2500]);
        subplot(2,2,4); semilogx(wvnrs,errs_phi./errs_phiBAE); title('Ratio_\phi'
```

```
            ); hold on; grid on;
    end
%     legend(strread(num2str(1:size(Ps,3)*size(Ps,2)),'%s'));
% legend('P=2 | nD=0','P=4 | nD=0','P=1 | nD=1','P=2 | nD=1','Location','
    NorthWest');
legend('P=2 | nD=0','P=4 | nD=0','P=3 | nD=0','P=5 | nD=0','P=1 | nD=1','P=2 | nD
    =1','Location','NorthWest');
    array2table([1:(size(Ps,2))*(size(Ps,3));HHs(1,:);nDerivs(1,:);Ps(1,:);nVars
        (1,:)],'RowNames',{'#','HH','nDeriv','P','nVars'})
end
if saveFig
    name = ['Plotjes/errors_Nwv',num2str(Nwv),'_theta',num2str(rad2deg(theta)),'_
        ',datestr(now,'yyyymmdd_HHMM')];
    savefig([name,'.fig']);
    print([name,'.png'],'-dpng');
end
% save(['Data/data_',datestr(now,'ddmmyy_HHMMSS'),'.mat'],'mesh','phi*','u*','v
    *','psi*','err*','wvnrs','indA','indB','indC','Ps','HHs','nDerivs','nVars');

%% tmp
if 1
figure;
for i=1:(indA-1)
    semilogx(wvnrs(:,:,i),errs(:,:,i)./errsBAE(:,:,i),'--x'); title('Ratio');
        hold on; grid on; ylim([0 15]); ylabel('DE/BAE'); xlabel('\omega'); xlim
        ([5 200]); grid minor
%     semilogx(wvnrs(:,:,i),errs_phi(:,:,i)./errs_phiBAE(:,:,i),'--x'); hold on;
    grid on; ylim([0 15]); ylabel('DE/BAE'); xlabel('\omega'); xlim([5 200]); grid
     minor
%     loglog(wvnrs(:,:,i),errs_phi(:,:,i),'--x'); title('\phi'); hold on; grid on
    ; ylabel('DE'); xlim([5 200]);
%     loglog(wvnrs(:,:,i),errs_ux(:,:,i),'--x'); title('ux'); hold on; grid on;
    ylabel('DE'); xlim([5 200]);
    legend('P=2 | nD=0','P=4 | nD=0','P=3 | nD=0','P=5 | nD=0','P=1 | nD=1','P=2
        | nD=1','Location','NorthWest');
end
hold off;
if 0
    name = ['Plotjes/Nieuw/errors_DEBAE_Nwv',num2str(Nwv),'_theta',num2str(
        rad2deg(theta)),'_',datestr(now,'yyyymmdd_HHMM')];
%     name = ['Plotjes/Nieuw/errors_DEBAEphi_Nwv',num2str(Nwv),'_theta',num2str(
    rad2deg(theta)),'_',datestr(now,'yyyymmdd_HHMM')];
%     name = ['Plotjes/Nieuw/errors_DEphi_Nwv',num2str(Nwv),'_theta',num2str(
    rad2deg(theta)),'_',datestr(now,'yyyymmdd_HHMM')];
%     name = ['Plotjes/Nieuw/errors_DEux_Nwv',num2str(Nwv),'_theta',num2str(
    rad2deg(theta)),'_',datestr(now,'yyyymmdd_HHMM')];
    savefig([name,'.fig']);
    print([name,'.png'],'-dpng');
end
end
```

B.3 mass_matrices2D_v4_2.m

```
function [M00,M11,M22,M02_H,M11_H,M1d1d] = mass_matrices2D_v4_2(P,N,M,nDeriv,
    domain) %
% Add storage of matrices
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% mass_matrices_v4.m
%
% N: Number of elements in horizontal direction in primal grid; using GLLnodes
%
% Polynomial order in primal grid is N & M
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

if 0
    close all;
    clearvars -except other
    clc
    N = 16; M=16; P=2; nDeriv=0;
    domain.x = [0 1];
    domain.y = [0 1];
else
    if nargin<5
        domain.x = [-1 1];
        domain.y = [-1 1];
        if nargin<4
            nDeriv=0;
            if nargin<3
                if nargin<2
                    N=1;
                end
                M=N;
            end
        end
    end
end

clrvrs = 0;

% location = 'D:\Stack\Documents\Studie\TU Delft\Master\Thesis\solvers\
    Hermite_solver_2D\mat\mass_matrices\';
% filename = ['matrices_P',num2str(P),'_N',num2str(N),'_M',num2str(M),'_nDeriv',
    num2str(nDeriv),'.mat'];
% if exist([location,filename],'file')
%     load([location,filename]);
% else

    Pf = 2*(P+1);
    dPf = 2/Pf;


    %% Nodes and weights
    xN = linspace(domain.x(1),domain.x(2),N+1);
    yN = linspace(domain.y(1),domain.y(2),M+1);
    % Primal Grid
```

```
% [int_nodes_xi,w_xi] = GLLnodes(P);
% Fine - for higher order integration
[int_nodes_xi_f,w_xi_f] = GLLnodes(Pf);
int_nodes_xi_f = (int_nodes_xi_f+1)*(domain.x(2)-domain.x(1))/2;
[int_nodes_eta_f,w_eta_f] = GLLnodes(Pf);
int_nodes_eta_f = (int_nodes_eta_f+1)*(domain.y(2)-domain.y(1))/2;
dx = diff(domain.x)/N;
dy = diff(domain.y)/M;
scale = dx*dy/4;


%% Nodal and edge basis functions
[h_xi_f,hp_xi_f,e_xi_f] = HermitePoly(int_nodes_xi_f/N,P,nDeriv,N*2/(domain.x
    (2)-domain.x(1)),0);
[h_eta_f,hp_eta_f,e_eta_f] = HermitePoly(int_nodes_eta_f/M,P,nDeriv,M*2/(
    domain.y(2)-domain.y(1)),0);
%     [h_xi_f,e_xi_f,hp_xi_f] = MimeticpolyVal(int_nodes_xi_f,P,1);
%     [h_eta_f,e_eta_f,hp_eta_f] = MimeticpolyVal(int_nodes_eta_f,P,1);


%% Number of nodes, edges and volumes
nn = (P+1)*(P+1);   % Number of nodes in an element
ne = P*(P+1)*2; % Number of edges in an element
nv = P*P;   % Number of volumes in an element
nn_g = (N*P+1)*(M*P+1); % Number of nodes globally
ne_g_h = N*P*(M*P+1);   % Number of horizontal edges globally
ne_g_v = M*P*(N*P+1);   % Number of vertical edges globally
ne_g = ne_g_h+ne_g_v;   % Number of edges globally
nv_g = (N*P)*(M*P); % Number of volumes globally

%% Mass matrix M00
h0h0_h0h0 = zeros(nn);
if nDeriv==1,
    h0h0_h1h0 = h0h0_h0h0; h0h0_h0h1=h0h0_h0h0; h0h0_h1h1=h0h0_h0h0;
    h1h0_h0h0 = zeros(nn); h1h0_h1h0=h0h0_h0h0; h1h0_h0h1=h0h0_h0h0;
        h1h0_h1h1=h0h0_h0h0;
    h0h1_h0h0 = zeros(nn); h0h1_h1h0=h0h0_h0h0; h0h1_h0h1=h0h0_h0h0;
        h0h1_h1h1=h0h0_h0h0;
    h1h1_h0h0 = zeros(nn); h1h1_h1h0=h0h0_h0h0; h1h1_h0h1=h0h0_h0h0;
        h1h1_h1h1=h0h0_h0h0;
end
for i=1:P+1
    for j=1:P+1
        localnode_i = i+(j-1)*(P+1);
        for k=1:P+1
            for l=1:P+1
                localnode_k = k+(l-1)*(P+1);
                h0h0_h0h0(localnode_i,localnode_k) = h0h0_h0h0(localnode_i,
                    localnode_k) + w_xi_f*((h_eta_f(j,:)'*h_xi_f(i,:)).*(
                    h_eta_f(l,:)'*h_xi_f(k,:)))*w_eta_f';
                if nDeriv==1
                    h0h0_h1h0(localnode_i,localnode_k) = h0h0_h1h0(
                        localnode_i,localnode_k) + w_xi_f*((h_eta_f(j,:)'*
                        h_xi_f(i,:)).*(h_eta_f(l,:)'*h_xi_f(k+(P+1),:)))*
                        w_eta_f';
                    h0h0_h0h1(localnode_i,localnode_k) = h0h0_h0h1(
                        localnode_i,localnode_k) + w_xi_f*((h_eta_f(j,:)'*
```

```
                        h_xi_f(i,:)).*(h_eta_f(l+(P+1),:)'*h_xi_f(k,:)))*
                        w_eta_f';
                    h0h0_h1h1(localnode_i,localnode_k) = h0h0_h1h1(
                        localnode_i,localnode_k) + w_xi_f*((h_eta_f(j,:)'*
                        h_xi_f(i,:)).*(h_eta_f(l+(P+1),:)'*h_xi_f(k+(P+1),:)))
                        *w_eta_f';

                    h1h0_h1h0(localnode_i,localnode_k) = h1h0_h1h0(
                        localnode_i,localnode_k) + w_xi_f*((h_eta_f(j,:)'*
                        h_xi_f(i+(P+1),:)).*(h_eta_f(l,:)'*h_xi_f(k+(P+1),:)))
                        *w_eta_f';
                    h1h0_h0h1(localnode_i,localnode_k) = h1h0_h0h1(
                        localnode_i,localnode_k) + w_xi_f*((h_eta_f(j,:)'*
                        h_xi_f(i+(P+1),:)).*(h_eta_f(l+(P+1),:)'*h_xi_f(k,:)))
                        *w_eta_f';
                    h1h0_h1h1(localnode_i,localnode_k) = h1h0_h1h1(
                        localnode_i,localnode_k) + w_xi_f*((h_eta_f(j,:)'*
                        h_xi_f(i+(P+1),:)).*(h_eta_f(l+(P+1),:)'*h_xi_f(k+(P
                        +1),:)))*w_eta_f';

                    h0h1_h0h1(localnode_i,localnode_k) = h0h1_h0h1(
                        localnode_i,localnode_k) + w_xi_f*((h_eta_f(j+(P+1),:)
                        '*h_xi_f(i,:)).*(h_eta_f(l+(P+1),:)'*h_xi_f(k,:)))*
                        w_eta_f';
                    h0h1_h1h1(localnode_i,localnode_k) = h0h1_h1h1(
                        localnode_i,localnode_k) + w_xi_f*((h_eta_f(j+(P+1),:)
                        '*h_xi_f(i,:)).*(h_eta_f(l+(P+1),:)'*h_xi_f(k+(P+1),:)
                        ))*w_eta_f';

                    h1h1_h1h1(localnode_i,localnode_k) = h1h1_h1h1(
                        localnode_i,localnode_k) + w_xi_f*((h_eta_f(j+(P+1),:)
                        '*h_xi_f(i+(P+1),:)).*(h_eta_f(l+(P+1),:)'*h_xi_f(k+(P
                        +1),:)))*w_eta_f';
                end
            end
        end
    end
end
if nDeriv==1,
    M00_l = [h0h0_h0h0 h0h0_h1h0 h0h0_h0h1 h0h0_h1h1;
        h0h0_h1h0' h1h0_h1h0 h1h0_h0h1 h1h0_h1h1;
        h0h0_h0h1' h1h0_h0h1' h0h1_h0h1 h0h1_h1h1;
        h0h0_h1h1' h1h0_h1h1' h0h1_h1h1' h1h1_h1h1]*scale;
else
    M00_l=h0h0_h0h0*scale;
end
if clrvrs, clearvars -except clrvrs P N M nDeriv dual M00_l nn ne nv nn_g
    ne_g nv_g w_xi_f w_eta_f h_xi_f hp_xi_f e_xi_f h_eta_f hp_eta_f e_eta_f
    scale m00 m11 m22 m02d m11d; end

%% M11
% horizontal edges
eh0_eh0 = zeros(ne/2);
if nDeriv==1,
    eh0_eh1=eh0_eh0; eh1_eh0=eh0_eh0; eh1_eh1=eh0_eh0;
    eh0_dh1h0 = zeros(ne/2,nn); eh0_dh1h1=eh0_dh1h0; eh1_dh1h0=eh0_dh1h0;
        eh1_dh1h1=eh0_dh1h0;
```

```
    end
    for j=1:P+1
        for i=1:P
            edge_i = i + (j-1)*P;
            % horizontal edges
            for l=1:P+1
                for k=1:P
                    edge_k = k + (l-1)*P;
                    eh0_eh0(edge_i,edge_k) = eh0_eh0(edge_i,edge_k) + w_xi_f*((
                        h_eta_f(j,:)'*e_xi_f(i,:)).*(h_eta_f(l,:)'*e_xi_f(k,:)))*
                        w_eta_f';
                    if nDeriv==1
                        eh0_eh1(edge_i,edge_k) = eh0_eh1(edge_i,edge_k) + w_xi_f
                            *((h_eta_f(j,:)'*e_xi_f(i,:)).*(h_eta_f(l+(P+1),:)'*
                            e_xi_f(k,:)))*w_eta_f';
                        eh1_eh0(edge_i,edge_k) = eh1_eh0(edge_i,edge_k) + w_xi_f
                            *((h_eta_f(j+(P+1),:)'*e_xi_f(i,:)).*(h_eta_f(l,:)'*
                            e_xi_f(k,:)))*w_eta_f';
                        eh1_eh1(edge_i,edge_k) = eh1_eh1(edge_i,edge_k) + w_xi_f
                            *((h_eta_f(j+(P+1),:)'*e_xi_f(i,:)).*(h_eta_f(l+(P+1)
                            ,:)'*e_xi_f(k,:)))*w_eta_f';
                    end
                end
            end
            if nDeriv==1
                % nodes
                for l=1:P+1
                    for k=1:P+1
                        node_k = k + (l-1)*(P+1);
                        eh0_dh1h0(edge_i,node_k) = eh0_dh1h0(edge_i,node_k) +
                            w_xi_f*((h_eta_f(j,:)'*e_xi_f(i,:)).*(h_eta_f(l,:)'*
                            hp_xi_f(k+(P+1),:)))*w_eta_f';
                        eh0_dh1h1(edge_i,node_k) = eh0_dh1h1(edge_i,node_k) +
                            w_xi_f*((h_eta_f(j,:)'*e_xi_f(i,:)).*(h_eta_f(l+(P+1)
                            ,:)'*hp_xi_f(k+(P+1),:)))*w_eta_f';
                        eh1_dh1h0(edge_i,node_k) = eh1_dh1h0(edge_i,node_k) +
                            w_xi_f*((h_eta_f(j+(P+1),:)'*e_xi_f(i,:)).*(h_eta_f(l
                            ,:)'*hp_xi_f(k+(P+1),:)))*w_eta_f';
                        eh1_dh1h1(edge_i,node_k) = eh1_dh1h1(edge_i,node_k) +
                            w_xi_f*((h_eta_f(j+(P+1),:)'*e_xi_f(i,:)).*(h_eta_f(l
                            +(P+1),:)'*hp_xi_f(k+(P+1),:)))*w_eta_f';
                    end
                end
            end
        end
    end
    if nDeriv==1
        % nodes
        dh1h0_eh0 = zeros(nn,ne/2); dh1h0_eh1=dh1h0_eh0; dh1h1_eh0=dh1h0_eh0;
            dh1h1_eh1=dh1h0_eh0;
        dh1h0_dh1h0 = zeros(nn); dh1h0_dh1h1=dh1h0_dh1h0; dh1h1_dh1h0=dh1h0_dh1h0
            ; dh1h1_dh1h1=dh1h0_dh1h0;
        for j=1:P+1
            for i=1:P+1
                node_i = i + (j-1)*(P+1);
                % horizontal edges
                for l=1:P+1
```

```
                for k=1:P
                    edge_k = k + (l-1)*P;
                    dh1h0_eh0(node_i,edge_k) = dh1h0_eh0(node_i,edge_k) +
                        w_xi_f*((h_eta_f(j,:)'*hp_xi_f(i+(P+1),:)).*(h_eta_f(l
                        ,:)'*e_xi_f(k,:)))*w_eta_f';
                    dh1h0_eh1(node_i,edge_k) = dh1h0_eh1(node_i,edge_k) +
                        w_xi_f*((h_eta_f(j,:)'*hp_xi_f(i+(P+1),:)).*(h_eta_f(l
                        +(P+1),:)'*e_xi_f(k,:)))*w_eta_f';
                    dh1h1_eh0(node_i,edge_k) = dh1h1_eh0(node_i,edge_k) +
                        w_xi_f*((h_eta_f(j+(P+1),:)'*hp_xi_f(i+(P+1),:)).*(
                        h_eta_f(l,:)'*e_xi_f(k,:)))*w_eta_f';
                    dh1h1_eh1(node_i,edge_k) = dh1h1_eh1(node_i,edge_k) +
                        w_xi_f*((h_eta_f(j+(P+1),:)'*hp_xi_f(i+(P+1),:)).*(
                        h_eta_f(l+(P+1),:)'*e_xi_f(k,:)))*w_eta_f';
                end
            end
            % nodes
            for l=1:P+1
                for k=1:P+1
                    node_k = k + (l-1)*(P+1);
                    dh1h0_dh1h0(node_i,node_k) = dh1h0_dh1h0(node_i,node_k) +
                         w_xi_f*((h_eta_f(j,:)'*hp_xi_f(i+(P+1),:)).*(h_eta_f(
                        l,:)'*hp_xi_f(k+(P+1),:)))*w_eta_f';
                    dh1h0_dh1h1(node_i,node_k) = dh1h0_dh1h1(node_i,node_k) +
                         w_xi_f*((h_eta_f(j,:)'*hp_xi_f(i+(P+1),:)).*(h_eta_f(
                        l+(P+1),:)'*hp_xi_f(k+(P+1),:)))*w_eta_f';
                    dh1h1_dh1h0(node_i,node_k) = dh1h1_dh1h0(node_i,node_k) +
                         w_xi_f*((h_eta_f(j+(P+1),:)'*hp_xi_f(i+(P+1),:)).*(
                        h_eta_f(l,:)'*hp_xi_f(k+(P+1),:)))*w_eta_f';
                    dh1h1_dh1h1(node_i,node_k) = dh1h1_dh1h1(node_i,node_k) +
                         w_xi_f*((h_eta_f(j+(P+1),:)'*hp_xi_f(i+(P+1),:)).*(
                        h_eta_f(l+(P+1),:)'*hp_xi_f(k+(P+1),:)))*w_eta_f';
                end
            end
        end
    end
end

% vertical edges
h0e_h0e = zeros(ne/2);
if nDeriv==1
    h0e_h1e = h0e_h0e; h1e_h0e=h0e_h0e; h1e_h1e=h0e_h0e;
    h0e_h0dh1 = zeros(ne/2,nn); h0e_h1dh1=h0e_h0dh1; h1e_h0dh1=h0e_h0dh1;
        h1e_h1dh1=h0e_h0dh1;
end
for j=1:P
    for i=1:P+1
        edge_i = i + (j-1)*(P+1);
        % vertical edges
        for l=1:P
            for k=1:P+1
                edge_k = k + (l-1)*(P+1);
                h0e_h0e(edge_i,edge_k) = h0e_h0e(edge_i,edge_k) + w_xi_f*((
                    e_eta_f(j,:)'*h_xi_f(i,:)).*(e_eta_f(l,:)'*h_xi_f(k,:)))*
                    w_eta_f';
                if nDeriv==1
                    h0e_h1e(edge_i,edge_k) = h0e_h1e(edge_i,edge_k) + w_xi_f
```

```
                        *((e_eta_f(j,:)'*h_xi_f(i,:)).*(e_eta_f(l,:)'*h_xi_f(k
                        +(P+1),:)))*w_eta_f';
                    h1e_h0e(edge_i,edge_k) = h1e_h0e(edge_i,edge_k) + w_xi_f
                        *((e_eta_f(j,:)'*h_xi_f(i+(P+1),:)).*(e_eta_f(l,:)'*
                        h_xi_f(k,:)))*w_eta_f';
                    h1e_h1e(edge_i,edge_k) = h1e_h1e(edge_i,edge_k) + w_xi_f
                        *((e_eta_f(j,:)'*h_xi_f(i+(P+1),:)).*(e_eta_f(l,:)'*
                        h_xi_f(k+(P+1),:)))*w_eta_f';
                end
            end
        end
        if nDeriv==1
            % nodes
            for l=1:P+1
                for k=1:P+1
                    node_k = k + (l-1)*(P+1);
                    h0e_h0dh1(edge_i,node_k) = h0e_h0dh1(edge_i,node_k) +
                        w_xi_f*((e_eta_f(j,:)'*h_xi_f(i,:)).*(hp_eta_f(l+(P+1)
                        ,:)'*h_xi_f(k,:)))*w_eta_f';
                    h0e_h1dh1(edge_i,node_k) = h0e_h1dh1(edge_i,node_k) +
                        w_xi_f*((e_eta_f(j,:)'*h_xi_f(i,:)).*(hp_eta_f(l+(P+1)
                        ,:)'*h_xi_f(k+(P+1),:)))*w_eta_f';
                    h1e_h0dh1(edge_i,node_k) = h1e_h0dh1(edge_i,node_k) +
                        w_xi_f*((e_eta_f(j,:)'*h_xi_f(i+(P+1),:)).*(hp_eta_f(l
                        +(P+1),:)'*h_xi_f(k,:)))*w_eta_f';
                    h1e_h1dh1(edge_i,node_k) = h1e_h1dh1(edge_i,node_k) +
                        w_xi_f*((e_eta_f(j,:)'*h_xi_f(i+(P+1),:)).*(hp_eta_f(l
                        +(P+1),:)'*h_xi_f(k+(P+1),:)))*w_eta_f';
                end
            end
        end
    end
end
if nDeriv==1
    % nodes
    h0dh1_h0e = zeros(nn,ne/2); h0dh1_h1e=h0dh1_h0e; h1dh1_h0e=h0dh1_h0e;
        h1dh1_h1e=h0dh1_h0e;
    h0dh1_h0dh1 = zeros(nn); h0dh1_h1dh1=h0dh1_h0dh1; h1dh1_h0dh1=h0dh1_h0dh1
        ; h1dh1_h1dh1=h0dh1_h0dh1;
    for j=1:P+1
        for i=1:P+1
            node_i = i + (j-1)*(P+1);
            % vertical edges
            for l=1:P
                for k=1:P+1
                    edge_k = k + (l-1)*(P+1);
                    h0dh1_h0e(node_i,edge_k) = h0dh1_h0e(node_i,edge_k) +
                        w_xi_f*((hp_eta_f(j+(P+1),:)'*h_xi_f(i,:)).*(e_eta_f(l
                        ,:)'*h_xi_f(k,:)))*w_eta_f';
                    h0dh1_h1e(node_i,edge_k) = h0dh1_h1e(node_i,edge_k) +
                        w_xi_f*((hp_eta_f(j+(P+1),:)'*h_xi_f(i,:)).*(e_eta_f(l
                        ,:)'*h_xi_f(k+(P+1),:)))*w_eta_f';
                    h1dh1_h0e(node_i,edge_k) = h1dh1_h0e(node_i,edge_k) +
                        w_xi_f*((hp_eta_f(j+(P+1),:)'*h_xi_f(i+(P+1),:)).*(
                        e_eta_f(l,:)'*h_xi_f(k,:)))*w_eta_f';
                    h1dh1_h1e(node_i,edge_k) = h1dh1_h1e(node_i,edge_k) +
                        w_xi_f*((hp_eta_f(j+(P+1),:)'*h_xi_f(i+(P+1),:)).*(
```

```
                        e_eta_f(l,:)'*h_xi_f(k+(P+1),:)))*w_eta_f';
                end
            end
            % nodes
            for l=1:P+1                                                    243
                for k=1:P+1
                    node_k = k + (l-1)*(P+1);
                    h0dh1_h0dh1(node_i,node_k) = h0dh1_h0dh1(node_i,node_k) +
                         w_xi_f*((hp_eta_f(j+(P+1),:)'*h_xi_f(i,:)).*(hp_eta_f
                        (l+(P+1),:)'*h_xi_f(k,:)))*w_eta_f';
                    h0dh1_h1dh1(node_i,node_k) = h0dh1_h1dh1(node_i,node_k) +
                         w_xi_f*((hp_eta_f(j+(P+1),:)'*h_xi_f(i,:)).*(hp_eta_f
                        (l+(P+1),:)'*h_xi_f(k+(P+1),:)))*w_eta_f';
                    h1dh1_h0dh1(node_i,node_k) = h1dh1_h0dh1(node_i,node_k) +   248
                         w_xi_f*((hp_eta_f(j+(P+1),:)'*h_xi_f(i+(P+1),:)).*(
                        hp_eta_f(l+(P+1),:)'*h_xi_f(k,:)))*w_eta_f';
                    h1dh1_h1dh1(node_i,node_k) = h1dh1_h1dh1(node_i,node_k) +
                         w_xi_f*((hp_eta_f(j+(P+1),:)'*h_xi_f(i+(P+1),:)).*(
                        hp_eta_f(l+(P+1),:)'*h_xi_f(k+(P+1),:)))*w_eta_f';
                end
            end
        end
    end                                                                    253
end

if nDeriv==1,
    M11x = [eh0_eh0 eh0_dh1h0 eh0_eh1 eh0_dh1h1;
        dh1h0_eh0 dh1h0_dh1h0 dh1h0_eh1 dh1h0_dh1h1;                       258
        eh1_eh0 eh1_dh1h0 eh1_eh1 eh1_dh1h1;
        dh1h1_eh0 dh1h1_dh1h0 dh1h1_eh1 dh1h1_dh1h1]*scale;
    M11y = [h0e_h0e h0e_h1e h0e_h0dh1 h0e_h1dh1;
        h1e_h0e h1e_h1e h1e_h0dh1 h1e_h1dh1;
        h0dh1_h0e h0dh1_h1e h0dh1_h0dh1 h0dh1_h1dh1;                       263
        h1dh1_h0e h1dh1_h1e h1dh1_h0dh1 h1dh1_h1dh1]*scale;
    M1d1d_l = blkdiag(M11y,M11x);
    M11_l = blkdiag(M11x,M11y);
else
    M1d1d_l=blkdiag(h0e_h0e,eh0_eh0)*scale;                                268
    M11_l=blkdiag(eh0_eh0,h0e_h0e)*scale;
end
if clrvrs, clearvars -except clrvrs P N M nDeriv dual nn ne nv nn_g ne_g nv_g
     w_xi_f w_eta_f h_xi_f hp_xi_f e_xi_f h_eta_f hp_eta_f e_eta_f scale M00_l
     M11_l m00 m11 m22 m02d m11d; end


                                                                           273
%% M22
% volume
ee_ee = zeros(nv); ee_dh1e = zeros(nv,ne/2); ee_edh1=ee_dh1e; ee_dh1dh1 =
    zeros(nv,nn);
for j=1:P
    for i=1:P                                                              278
        vol_i = i + (j-1)*P;
        % volume
        for l=1:P
            for k=1:P
                vol_k = k + (l-1)*P;                                       283
                ee_ee(vol_i,vol_k) = ee_ee(vol_i,vol_k) + w_xi_f*((e_eta_f(j
```

```
                    ,:)'*e_xi_f(i,:)).*(e_eta_f(l,:)'*e_xi_f(k,:)))*w_eta_f';
            end
        end
        if nDeriv==1
            % vertical edge                                                    288
            for l=1:P
                for k=1:P+1
                    edge_k = k + (l-1)*(P+1);
                    ee_dh1e(vol_i,edge_k) = ee_dh1e(vol_i,edge_k) + w_xi_f*((
                        e_eta_f(j,:)'*e_xi_f(i,:)).*(e_eta_f(l,:)'*hp_xi_f(k+(
                        P+1),:)))*w_eta_f';
                end                                                           293
            end
            % horizontal edge
            for l=1:P+1
                for k=1:P
                    edge_k = k + (l-1)*P;                                     298
                    ee_edh1(vol_i,edge_k) = ee_edh1(vol_i,edge_k) + w_xi_f*((
                        e_eta_f(j,:)'*e_xi_f(i,:)).*(hp_eta_f(l+(P+1),:)'*
                        e_xi_f(k,:)))*w_eta_f';
                end
            end
            % node
            for l=1:P+1                                                       303
                for k=1:P+1
                    node_k = k + (l-1)*(P+1);
                    ee_dh1dh1(vol_i,node_k) = ee_dh1dh1(vol_i,node_k) +
                        w_xi_f*((e_eta_f(j,:)'*e_xi_f(i,:)).*(hp_eta_f(l+(P+1)
                        ,:)'*hp_xi_f(k+(P+1),:)))*w_eta_f';
                end
            end                                                               308
        end
    end
end
if nDeriv==1
    % vertical edge                                                          313
    dh1e_ee = zeros(ne/2,nv); dh1e_dh1e = zeros(ne/2); dh1e_edh1=dh1e_dh1e;
        dh1e_dh1dh1 = zeros(ne/2,nn);
    for j=1:P
        for i=1:P+1
            edge_i = i + (j-1)*(P+1);
            % volume                                                          318
            for l=1:P
                for k=1:P
                    vol_k = k + (l-1)*P;
                    dh1e_ee(edge_i,vol_k) = dh1e_ee(edge_i,vol_k) + w_xi_f*((
                        e_eta_f(j,:)'*hp_xi_f(i+(P+1),:)).*(e_eta_f(l,:)'*
                        e_xi_f(k,:)))*w_eta_f';
                end                                                           323
            end
            % vertical edge
            for l=1:P
                for k=1:P+1
                    edge_k = k + (l-1)*(P+1);                                 328
                    dh1e_dh1e(edge_i,edge_k) = dh1e_dh1e(edge_i,edge_k) +
                        w_xi_f*((e_eta_f(j,:)'*hp_xi_f(i+(P+1),:)).*(e_eta_f(l
                        ,:)'*hp_xi_f(k+(P+1),:)))*w_eta_f';
```

```
            end
        end
        % horizontal edge
        for l=1:P+1
            for k=1:P
                edge_k = k + (l-1)*P;
                dh1e_edh1(edge_i,edge_k) = dh1e_edh1(edge_i,edge_k) +
                    w_xi_f*((e_eta_f(j,:)'*hp_xi_f(i+(P+1),:)).*(hp_eta_f(
                    l+(P+1),:)'*e_xi_f(k,:)))*w_eta_f';
            end
        end
        % node
        for l=1:P+1
            for k=1:P+1
                node_k = k + (l-1)*(P+1);
                dh1e_dh1dh1(edge_i,node_k) = dh1e_dh1dh1(edge_i,node_k) +
                     w_xi_f*((e_eta_f(j,:)'*hp_xi_f(i+(P+1),:)).*(hp_eta_f
                    (l+(P+1),:)'*hp_xi_f(k+(P+1),:)))*w_eta_f';
            end
        end
    end
end
% horizontal edge
edh1_ee = zeros(ne/2,nv); edh1_dh1e = zeros(ne/2); edh1_edh1 = zeros(ne
    /2); edh1_dh1dh1 = zeros(ne/2,nn);
for j=1:P+1
    for i=1:P
        edge_i = i + (j-1)*P;
        % volume
        for l=1:P
            for k=1:P
                vol_k = k + (l-1)*P;
                edh1_ee(edge_i,vol_k) = edh1_ee(edge_i,vol_k) + w_xi_f*((
                    hp_eta_f(j+(P+1),:)'*e_xi_f(i,:)).*(e_eta_f(l,:)'*
                    e_xi_f(k,:)))*w_eta_f';
            end
        end
        % vertical edge
        for l=1:P
            for k=1:P+1
                edge_k = k + (l-1)*(P+1);
                edh1_dh1e(edge_i,edge_k) = edh1_dh1e(edge_i,edge_k) +
                    w_xi_f*((hp_eta_f(j+(P+1),:)'*e_xi_f(i,:)).*(e_eta_f(l
                    ,:)'*hp_xi_f(k+(P+1),:)))*w_eta_f';
            end
        end
        % horizontal edge
        for l=1:P+1
            for k=1:P
                edge_k = k + (l-1)*P;
                edh1_edh1(edge_i,edge_k) = edh1_edh1(edge_i,edge_k) +
                    w_xi_f*((hp_eta_f(j+(P+1),:)'*e_xi_f(i,:)).*(hp_eta_f(
                    l+(P+1),:)'*e_xi_f(k,:)))*w_eta_f';
            end
        end
        % node
        for l=1:P+1
```

```
                for k=1:P+1
                    node_k = k + (l-1)*(P+1);
                    edh1_dh1dh1(edge_i,node_k) = edh1_dh1dh1(edge_i,node_k) +
                        w_xi_f*((hp_eta_f(j+(P+1),:)'*e_xi_f(i,:)).*(hp_eta_f
                        (l+(P+1),:)'*hp_xi_f(k+(P+1),:)))*w_eta_f';
                end
            end
        end
    end
    % nodes
    dh1dh1_ee = zeros(nn,nv); dh1dh1_dh1e = zeros(nn,ne/2); dh1dh1_edh1 =
        zeros(nn,ne/2); dh1dh1_dh1dh1 = zeros(nn);
    for j=1:P+1
        for i=1:P+1
            node_i = i + (j-1)*(P+1);
            % volume
            for l=1:P
                for k=1:P
                    vol_k = k + (l-1)*P;
                    dh1dh1_ee(node_i,vol_k) = dh1dh1_ee(node_i,vol_k) +
                        w_xi_f*((hp_eta_f(j+(P+1),:)'*hp_xi_f(i+(P+1),:)).*(
                        e_eta_f(l,:)'*e_xi_f(k,:)))*w_eta_f';
                end
            end
            % vertical edge
            for l=1:P
                for k=1:P+1
                    edge_k = k + (l-1)*(P+1);
                    dh1dh1_dh1e(node_i,edge_k) = dh1dh1_dh1e(node_i,edge_k) +
                        w_xi_f*((hp_eta_f(j+(P+1),:)'*hp_xi_f(i+(P+1),:)).*(
                        e_eta_f(l,:)'*hp_xi_f(k+(P+1),:)))*w_eta_f';
                end
            end
            % horizontal edge
            for l=1:P+1
                for k=1:P
                    edge_k = k + (l-1)*P;
                    dh1dh1_edh1(node_i,edge_k) = dh1dh1_edh1(node_i,edge_k) +
                        w_xi_f*((hp_eta_f(j+(P+1),:)'*hp_xi_f(i+(P+1),:)).*(
                        hp_eta_f(l+(P+1),:)'*e_xi_f(k,:)))*w_eta_f';
                end
            end
            % node
            for l=1:P+1
                for k=1:P+1
                    node_k = k + (l-1)*(P+1);
                    dh1dh1_dh1dh1(node_i,node_k) = dh1dh1_dh1dh1(node_i,
                        node_k) + w_xi_f*((hp_eta_f(j+(P+1),:)'*hp_xi_f(i+(P
                        +1),:)).*(hp_eta_f(l+(P+1),:)'*hp_xi_f(k+(P+1),:)))*
                        w_eta_f';
                end
            end
        end
    end
end

if nDeriv==1,
```

```
    M22_l = [ee_ee ee_dh1e ee_edh1 ee_dh1dh1;
        dh1e_ee dh1e_dh1e dh1e_edh1 dh1e_dh1dh1;
        edh1_ee edh1_dh1e edh1_edh1 edh1_dh1dh1;
        dh1dh1_ee dh1dh1_dh1e dh1dh1_edh1 dh1dh1_dh1dh1]*scale;
else
    M22_l=ee_ee*scale;
end
if clrvrs, clearvars -except clrvrs P N M nDeriv dual nn ne nv nn_g ne_g nv_g
     w_xi_f w_eta_f h_xi_f hp_xi_f e_xi_f h_eta_f hp_eta_f e_eta_f scale M00_l
     M11_l M22_l m00 m11 m22 m02d m11d; end


%% Hodge matrix - M02
% node-vol/edge/node
h0h0_ee = zeros(nn,nv);
if nDeriv==1
    h1h0_ee=h0h0_ee;  h0h1_ee=h0h0_ee;  h1h1_ee=h0h0_ee;
    h0h0_dh1e = zeros(nn,ne/2); h1h0_dh1e=h0h0_dh1e;  h0h1_dh1e=h0h0_dh1e;
        h1h1_dh1e=h0h0_dh1e;
    h0h0_edh1 = zeros(nn,ne/2); h1h0_edh1=h0h0_edh1;  h0h1_edh1=h0h0_edh1;
        h1h1_edh1=h0h0_edh1;
    h0h0_dh1dh1 = zeros(nn); h1h0_dh1dh1=h0h0_dh1dh1; h0h1_dh1dh1=h0h0_dh1dh1
        ; h1h1_dh1dh1=h0h0_dh1dh1;
end
for j=1:P+1
    for i=1:P+1
        node_i = i + (j-1)*(P+1);
        % volumes
        for l=1:P
            for k=1:P
                vol_k = k + (l-1)*P;
                h0h0_ee(node_i,vol_k) = h0h0_ee(node_i,vol_k) + w_xi_f*((
                    h_eta_f(j,:)'*h_xi_f(i,:)).*(e_eta_f(l,:)'*e_xi_f(k,:)))*
                    w_eta_f';
                if nDeriv==1
                    h1h0_ee(node_i,vol_k) = h1h0_ee(node_i,vol_k) + w_xi_f*((
                        h_eta_f(j,:)'*h_xi_f(i+(P+1),:)).*(e_eta_f(l,:)'*
                        e_xi_f(k,:)))*w_eta_f';
                    h0h1_ee(node_i,vol_k) = h0h1_ee(node_i,vol_k) + w_xi_f*((
                        h_eta_f(j+(P+1),:)'*h_xi_f(i,:)).*(e_eta_f(l,:)'*
                        e_xi_f(k,:)))*w_eta_f';
                    h1h1_ee(node_i,vol_k) = h1h1_ee(node_i,vol_k) + w_xi_f*((
                        h_eta_f(j+(P+1),:)'*h_xi_f(i+(P+1),:)).*(e_eta_f(l,:)
                        '*e_xi_f(k,:)))*w_eta_f';
                end
            end
        end
        if nDeriv==1
            % vertical edges
            for l=1:P
                for k=1:P+1
                    edge_k = k + (l-1)*(P+1);
                    h0h0_dh1e(node_i,edge_k) = h0h0_dh1e(node_i,edge_k) +
                        w_xi_f*((h_eta_f(j,:)'*h_xi_f(i,:)).*(e_eta_f(l,:)'*
                        hp_xi_f(k+(P+1),:)))*w_eta_f';
                    h1h0_dh1e(node_i,edge_k) = h1h0_dh1e(node_i,edge_k) +
                        w_xi_f*((h_eta_f(j,:)'*h_xi_f(i+(P+1),:)).*(e_eta_f(l
```

```
                    ,:)'*hp_xi_f(k+(P+1),:)))*w_eta_f';
                h0h1_dh1e(node_i,edge_k) = h0h1_dh1e(node_i,edge_k) +
                    w_xi_f*((h_eta_f(j+(P+1),:)'*h_xi_f(i,:)).*(e_eta_f(l
                    ,:)'*hp_xi_f(k+(P+1),:)))*w_eta_f';
                h1h1_dh1e(node_i,edge_k) = h1h1_dh1e(node_i,edge_k) +
                    w_xi_f*((h_eta_f(j+(P+1),:)'*h_xi_f(i+(P+1),:)).*(
                    e_eta_f(l,:)'*hp_xi_f(k+(P+1),:)))*w_eta_f';
            end
        end
        % horizontal edges
        for l=1:P+1
            for k=1:P
                edge_k = k + (l-1)*P;
                h0h0_edh1(node_i,edge_k) = h0h0_edh1(node_i,edge_k) +
                    w_xi_f*((h_eta_f(j,:)'*h_xi_f(i,:)).*(hp_eta_f(l+(P+1)
                    ,:)'*e_xi_f(k,:)))*w_eta_f';
                h1h0_edh1(node_i,edge_k) = h1h0_edh1(node_i,edge_k) +
                    w_xi_f*((h_eta_f(j,:)'*h_xi_f(i+(P+1),:)).*(hp_eta_f(l
                    +(P+1),:)'*e_xi_f(k,:)))*w_eta_f';
                h0h1_edh1(node_i,edge_k) = h0h1_edh1(node_i,edge_k) +
                    w_xi_f*((h_eta_f(j+(P+1),:)'*h_xi_f(i,:)).*(hp_eta_f(l
                    +(P+1),:)'*e_xi_f(k,:)))*w_eta_f';
                h1h1_edh1(node_i,edge_k) = h1h1_edh1(node_i,edge_k) +
                    w_xi_f*((h_eta_f(j+(P+1),:)'*h_xi_f(i+(P+1),:)).*(
                    hp_eta_f(l+(P+1),:)'*e_xi_f(k,:)))*w_eta_f';
            end
        end
        % nodes
        for l=1:P+1
            for k=1:P+1
                node_k = k + (l-1)*(P+1);
                h0h0_dh1dh1(node_i,node_k) = h0h0_dh1dh1(node_i,node_k) +
                     w_xi_f*((h_eta_f(j,:)'*h_xi_f(i,:)).*(hp_eta_f(l+(P
                    +1),:)'*hp_xi_f(k+(P+1),:)))*w_eta_f';
                h1h0_dh1dh1(node_i,node_k) = h1h0_dh1dh1(node_i,node_k) +
                     w_xi_f*((h_eta_f(j,:)'*h_xi_f(i+(P+1),:)).*(hp_eta_f(
                    l+(P+1),:)'*hp_xi_f(k+(P+1),:)))*w_eta_f';
                h0h1_dh1dh1(node_i,node_k) = h0h1_dh1dh1(node_i,node_k) +
                     w_xi_f*((h_eta_f(j+(P+1),:)'*h_xi_f(i,:)).*(hp_eta_f(
                    l+(P+1),:)'*hp_xi_f(k+(P+1),:)))*w_eta_f';
                h1h1_dh1dh1(node_i,node_k) = h1h1_dh1dh1(node_i,node_k) +
                     w_xi_f*((h_eta_f(j+(P+1),:)'*h_xi_f(i+(P+1),:)).*(
                    hp_eta_f(l+(P+1),:)'*hp_xi_f(k+(P+1),:)))*w_eta_f';
            end
        end
    end
end

if nDeriv==1,
    M02_H_l = [h0h0_ee h0h0_dh1e h0h0_edh1 h0h0_dh1dh1;
        h1h0_ee h1h0_dh1e h1h0_edh1 h1h0_dh1dh1;
        h0h1_ee h0h1_dh1e h0h1_edh1 h0h1_dh1dh1;
        h1h1_ee h1h1_dh1e h1h1_edh1 h1h1_dh1dh1]*scale;
else
    M02_H_l=h0h0_ee*scale;
end
```

```
if clrvrs, clearvars -except clrvrs P N M nDeriv dual nn ne nv nn_g ne_g nv_g
    w_xi_f w_eta_f h_xi_f hp_xi_f e_xi_f h_eta_f hp_eta_f e_eta_f scale M00_l
    M11_l M22_l M02_H_l m00 m11 m22 m02d m11d; end


%% Hodge matrix - M11
% horizontal edge
eh0_h0e = zeros(ne/2);
if nDeriv==1
    eh0_h1e=eh0_h0e; eh1_h0e=eh0_h0e; eh1_h1e=eh0_h0e;
    eh0_h0dh1 = zeros(ne/2,nn); eh0_h1dh1=eh0_h0dh1;  eh1_h0dh1=eh0_h0dh1;
        eh1_h1dh1=eh0_h0dh1;
end
for j=1:P+1
    for i=1:P
        edge_i = i + (j-1)*P;
        % vertical edge
        for l=1:P
            for k=1:P+1
                edge_k = k + (l-1)*(P+1);
                eh0_h0e(edge_i,edge_k) = eh0_h0e(edge_i,edge_k) + w_xi_f*((
                    h_eta_f(j,:)'*e_xi_f(i,:)).*(e_eta_f(l,:)'*h_xi_f(k,:)))*
                    w_eta_f';
                if nDeriv==1
                    eh0_h1e(edge_i,edge_k) = eh0_h1e(edge_i,edge_k) + w_xi_f
                        *((h_eta_f(j,:)'*e_xi_f(i,:)).*(e_eta_f(l,:)'*h_xi_f(k
                        +(P+1),:)))*w_eta_f';
                    eh1_h0e(edge_i,edge_k) = eh1_h0e(edge_i,edge_k) + w_xi_f
                        *((h_eta_f(j+(P+1),:)'*e_xi_f(i,:)).*(e_eta_f(l,:)'*
                        h_xi_f(k,:)))*w_eta_f';
                    eh1_h1e(edge_i,edge_k) = eh1_h1e(edge_i,edge_k) + w_xi_f
                        *((h_eta_f(j+(P+1),:)'*e_xi_f(i,:)).*(e_eta_f(l,:)'*
                        h_xi_f(k+(P+1),:)))*w_eta_f';
                end
            end
        end
        if nDeriv==1
            % node
            for l=1:P+1
                for k=1:P+1
                    node_k = k + (l-1)*(P+1);
                    eh0_h0dh1(edge_i,node_k) = eh0_h0dh1(edge_i,node_k) +
                        w_xi_f*((h_eta_f(j,:)'*e_xi_f(i,:)).*(hp_eta_f(l+(P+1)
                        ,:)'*h_xi_f(k,:)))*w_eta_f';
                    eh0_h1dh1(edge_i,node_k) = eh0_h1dh1(edge_i,node_k) +
                        w_xi_f*((h_eta_f(j,:)'*e_xi_f(i,:)).*(hp_eta_f(l+(P+1)
                        ,:)'*h_xi_f(k+(P+1),:)))*w_eta_f';
                    eh1_h0dh1(edge_i,node_k) = eh1_h0dh1(edge_i,node_k) +
                        w_xi_f*((h_eta_f(j+(P+1),:)'*e_xi_f(i,:)).*(hp_eta_f(l
                        +(P+1),:)'*h_xi_f(k,:)))*w_eta_f';
                    eh1_h1dh1(edge_i,node_k) = eh1_h1dh1(edge_i,node_k) +
                        w_xi_f*((h_eta_f(j+(P+1),:)'*e_xi_f(i,:)).*(hp_eta_f(l
                        +(P+1),:)'*h_xi_f(k+(P+1),:)))*w_eta_f';
                end
            end
        end
    end
```

```
end
% nodes
if nDeriv==1                                                                    538
    dh1h0_h0e = zeros(nn,ne/2); dh1h0_h1e=dh1h0_h0e; dh1h1_h0e=dh1h0_h0e;
        dh1h1_h1e=dh1h0_h0e;
    dh1h0_h0dh1 = zeros(nn); dh1h0_h1dh1=dh1h0_h0dh1;  dh1h1_h0dh1=
        dh1h0_h0dh1;  dh1h1_h1dh1=dh1h0_h0dh1;
    for j=1:P+1
        for i=1:P+1
            node_i = i + (j-1)*(P+1);                                           543
            % vertical edge
            for l=1:P
                for k=1:P+1
                    edge_k = k + (l-1)*(P+1);
                    dh1h0_h0e(node_i,edge_k) = dh1h0_h0e(node_i,edge_k) +        548
                        w_xi_f*((h_eta_f(j,:)'*hp_xi_f(i+(P+1),:)).*(e_eta_f(l
                        ,:)'*h_xi_f(k,:)))*w_eta_f';
                    dh1h0_h1e(node_i,edge_k) = dh1h0_h1e(node_i,edge_k) +
                        w_xi_f*((h_eta_f(j,:)'*hp_xi_f(i+(P+1),:)).*(e_eta_f(l
                        ,:)'*h_xi_f(k+(P+1),:)))*w_eta_f';
                    dh1h1_h0e(node_i,edge_k) = dh1h1_h0e(node_i,edge_k) +
                        w_xi_f*((h_eta_f(j+(P+1),:)'*hp_xi_f(i+(P+1),:)).*(
                        e_eta_f(l,:)'*h_xi_f(k,:)))*w_eta_f';
                    dh1h1_h1e(node_i,edge_k) = dh1h1_h1e(node_i,edge_k) +
                        w_xi_f*((h_eta_f(j+(P+1),:)'*hp_xi_f(i+(P+1),:)).*(
                        e_eta_f(l,:)'*h_xi_f(k+(P+1),:)))*w_eta_f';
                end
            end                                                                 553
            % node
            for l=1:P+1
                for k=1:P+1
                    node_k = k + (l-1)*(P+1);
                    dh1h0_h0dh1(node_i,node_k) = dh1h0_h0dh1(node_i,node_k) +    558
                         w_xi_f*((h_eta_f(j,:)'*hp_xi_f(i+(P+1),:)).*(hp_eta_f
                        (l+(P+1),:)'*h_xi_f(k,:)))*w_eta_f';
                    dh1h0_h1dh1(node_i,node_k) = dh1h0_h1dh1(node_i,node_k) +
                         w_xi_f*((h_eta_f(j,:)'*hp_xi_f(i+(P+1),:)).*(hp_eta_f
                        (l+(P+1),:)'*h_xi_f(k+(P+1),:)))*w_eta_f';
                    dh1h1_h0dh1(node_i,node_k) = dh1h1_h0dh1(node_i,node_k) +
                         w_xi_f*((h_eta_f(j+(P+1),:)'*hp_xi_f(i+(P+1),:)).*(
                        hp_eta_f(l+(P+1),:)'*h_xi_f(k,:)))*w_eta_f';
                    dh1h1_h1dh1(node_i,node_k) = dh1h1_h1dh1(node_i,node_k) +
                         w_xi_f*((h_eta_f(j+(P+1),:)'*hp_xi_f(i+(P+1),:)).*(
                        hp_eta_f(l+(P+1),:)'*h_xi_f(k+(P+1),:)))*w_eta_f';
                end
            end                                                                 563
        end
    end
end

if nDeriv==1,                                                                   568
    M11x_H = [eh0_h0e eh0_h1e eh0_h0dh1 eh0_h1dh1;
        dh1h0_h0e dh1h0_h1e dh1h0_h0dh1 dh1h0_h1dh1;
        eh1_h0e eh1_h1e eh1_h0dh1 eh1_h1dh1;
        dh1h1_h0e dh1h1_h1e dh1h1_h0dh1 dh1h1_h1dh1]*scale;
    M11_H_l = blkdiag(M11x_H,M11x_H');                                          573
else
```

```
        M11_H_l=blkdiag(eh0_h0e,eh0_h0e')*scale;
    end
    if clrvrs, clearvars -except clrvrs P N M nDeriv dual nn ne nv nn_g ne_g nv_g
         w_xi_f w_eta_f h_xi_f hp_xi_f e_xi_f h_eta_f hp_eta_f e_eta_f scale M00_l
         M11_l M22_l M02_H_l M11_H_l m00 m11 m22 m02d m11d; end

    %% Construct Global
%     if nDeriv==1
% %         M22 = sparse(zeros(nn_g+nv_g+ne_g)); M11 = sparse(zeros(2*(nn_g*2+
    ne_g))); M00 = sparse(zeros(nn_g*4)); M02_H = sparse(zeros(nn_g*4,nn_g+nv_g+
    ne_g)); M11_H = sparse(zeros(2*(nn_g*2+ne_g)));
%         M22 = sparse((nn_g+nv_g+ne_g),(nn_g+nv_g+ne_g),0); M11 = sparse((2*(
    nn_g*2+ne_g)),(2*(nn_g*2+ne_g)),0); M00 = sparse((nn_g*4),(nn_g*4),0); M02_H =
     sparse(nn_g*4,nn_g+nv_g+ne_g,0); M11_H = sparse((2*(nn_g*2+ne_g)),(2*(nn_g*2+
    ne_g)),0);
%     else
% %         M22 = sparse(zeros(nv_g)); M11 = sparse(zeros(ne_g)); M00 = sparse(
    zeros(nn_g)); M02_H = sparse(zeros(nn_g,nv_g)); M11_H = sparse(zeros(ne_g));
%         M22 = sparse(nv_g,nv_g,0); M11 = sparse(ne_g,ne_g,0); M00 = sparse(nn_g
    ,nn_g,0); M02_H = sparse(nn_g,nv_g,0); M11_H = sparse(ne_g,ne_g,0);
%     end

    if N*M>1
        [tmpi,tmpj,tmp_v] = find(M00_l); temp_v00 = zeros(numel(M00_l),1);
            temp_v00(tmpi+(tmpj-1)*max(tmpi),1) = tmp_v; clear tmp_v tmpi tmpj
        [tmpi,tmpj,tmp_v] = find(M11_l(1:end/2,1:end/2)); temp_v111 = zeros(numel
            (M11_l)/4,1); temp_v111(tmpi+(tmpj-1)*max(tmpi),1) = tmp_v; clear
            tmp_v tmpi tmpj
        [tmpi,tmpj,tmp_v] = find(M11_l(end/2+1:end,end/2+1:end)); temp_v11 =
            zeros(numel(M11_l)/4,1); temp_v11(tmpi+(tmpj-1)*max(tmpi),1) = tmp_v;
            clear tmp_v tmpi tmpj
        temp_v11 = [temp_v111;temp_v11]; clear temp_v111
        [tmpi,tmpj,tmp_v] = find(M1d1d_l(1:end/2,1:end/2)); temp_v1d1d1 = zeros(
            numel(M1d1d_l)/4,1); temp_v1d1d1(tmpi+(tmpj-1)*max(tmpi),1) = tmp_v;
            clear tmp_v tmpi tmpj
        [tmpi,tmpj,tmp_v] = find(M1d1d_l(end/2+1:end,end/2+1:end)); temp_v1d1d =
            zeros(numel(M1d1d_l)/4,1); temp_v1d1d(tmpi+(tmpj-1)*max(tmpi),1) =
            tmp_v; clear tmp_v tmpi tmpj
        temp_v1d1d = [temp_v1d1d1;temp_v1d1d]; clear temp_v1d1d1
        [tmpi,tmpj,tmp_v] = find(M22_l); temp_v22 = zeros(numel(M22_l),1);
            temp_v22(tmpi+(tmpj-1)*max(tmpi),1) = tmp_v; clear tmp_v tmpi tmpj
        [tmpi,tmpj,tmp_v] = find(M11_H_l(1:end/2,1:end/2)); temp_v11h1 = zeros(
            numel(M11_H_l)/4,1); temp_v11h1(tmpi+(tmpj-1)*max(tmpi),1) = tmp_v;
            clear tmp_v tmpi tmpj
        [tmpi,tmpj,tmp_v] = find(M11_H_l(end/2+1:end,end/2+1:end)); temp_v11h =
            zeros(numel(M11_H_l)/4,1); temp_v11h(tmpi+(tmpj-1)*max(tmpi),1) =
            tmp_v; clear tmp_v tmpi tmpj
        temp_v11h = [temp_v11h1;temp_v11h]; clear temp_v11h1
        [tmpi,tmpj,tmp_v] = find(M02_H_l); temp_v02h = zeros(numel(M02_H_l),1);
            temp_v02h(tmpi+(tmpj-1)*max(tmpi),1) = tmp_v; clear tmp_v tmpi tmpj

        i_M00=zeros(N*M*((nDeriv+1)^2*nn)^2,1);j_M00=i_M00;v_M00=i_M00; count00 =
             1;
        i_M11=zeros(2*N*M*(ne/2+nDeriv*(2*nn+ne/2))^2,1);j_M11=i_M11;v_M11=i_M11;
             count11 = 1;
        i_M1d1d=zeros(2*N*M*(ne/2+nDeriv*(2*nn+ne/2))^2,1);j_M1d1d=i_M1d1d;
```

```
    v_M1d1d=i_M1d1d; count1d1d = 1;
i_M22=zeros(N*M*(nv+nDeriv*(ne+nn))^2,1);j_M22=i_M22;v_M22=i_M22; count22
    = 1;
i_M02h=zeros(N*M*((nDeriv+1)^2*nn)*(nv+nDeriv*(ne+nn)),1);j_M02h=i_M02h;
    v_M02h=i_M02h; count02h = 1;
i_M11h=zeros(2*N*M*(ne/2+nDeriv*(2*nn+ne/2))^2,1);j_M11h=i_M11h;v_M11h=
    i_M11h; count11h = 1;
for iN=1:N
    for iM=1:M
        [lg_n,lg_e,lg_v] = local_globalv6(P,N,M,iN,iM,0);
        if nDeriv==1
            % [nodes; nodes; nodes; nodes]
            ind_g_00 = [lg_n(:,2);lg_n(:,2)+nn_g;lg_n(:,2)+2*nn_g;lg_n
                (:,2)+3*nn_g];
            ind_l_00 = [lg_n(:,1);lg_n(:,1)+nn;lg_n(:,1)+2*nn;lg_n(:,1)
                +3*nn];
            % [horizontal_edges; nodes; horizontal_edges; nodes;
                vertical_edges; vertical_edges; nodes; nodes;]
            ind_g_11p = [lg_e(1:end/2,2); lg_n(:,2)+ne_g_h;
                lg_e(1:end/2,2)+ne_g_h+nn_g; lg_n(:,2)+2*ne_g_h+nn_g;
                lg_e(end/2+1:end,2)+ne_g_h+2*nn_g; lg_e(end/2+1:end,2)+
                    ne_g+2*nn_g;
                lg_n(:,2)+2*ne_g+2*nn_g; lg_n(:,2)+2*ne_g+3*nn_g];
            ind_l_11p = [lg_e(1:end/2,1); lg_n(:,1)+ne/2;
                lg_e(1:end/2,1)+ne/2+nn; lg_n(:,1)+ne+nn;
                lg_e(end/2+1:end,1)+ne/2+2*nn; lg_e(end/2+1:end,1)+ne+2*
                    nn;
                lg_n(:,1)+2*ne+2*nn; lg_n(:,1)+2*ne+3*nn];
            % [vertical_edges; vertical_edges; nodes; nodes;
                horizontal_edges; nodes; horizontal_edges; nodes;]
            ind_g_11d = [lg_e(end/2+1:end,2)-ne_g_h; lg_e(end/2+1:end,2)-
                ne_g_h+ne_g_v;
                lg_n(:,2)+2*ne_g_v; lg_n(:,2)+2*ne_g_v+nn_g;
                lg_e(1:end/2,2)+2*ne_g_v+2*nn_g; lg_n(:,2)+ne_g+ne_g_v+2*
                    nn_g;
                lg_e(1:end/2,2)+ne_g+ne_g_v+3*nn_g; lg_n(:,2)+2*ne_g+3*
                    nn_g];
            ind_l_11d = [lg_e(end/2+1:end,1)-ne/2; lg_e(end/2+1:end,1);
                lg_n(:,1)+ne; lg_n(:,1)+ne+nn;
                lg_e(1:end/2,1)+ne+2*nn; lg_n(:,1)+3*ne/2+2*nn;
                lg_e(1:end/2,1)+3*ne/2+3*nn; lg_n(:,1)+2*ne+3*nn];
            ind_g_11 = ind_g_11p;
            ind_l_11 = ind_l_11p;
            ind_g_1d1d = ind_g_11d;
            ind_l_1d1d = ind_l_11d;
            % [volumes; vertical_edges; horizontal_edges; nodes]
            ind_g_22 = [lg_v(:,2);lg_e(end/2+1:end,2)-ne_g_h+nv_g;lg_e(1
                :end/2,2)+nv_g+ne_g_v;lg_n(:,2)+nv_g+ne_g];
            ind_l_22 = [lg_v(:,1);lg_e(end/2+1:end,1)-ne/2+nv;lg_e(1:end
                /2,1)+nv+ne/2;lg_n(:,1)+nv+ne];
        else
            ind_g_00 = lg_n(:,2);
            ind_l_00 = lg_n(:,1);
            ind_g_11p = lg_e(:,2);
            ind_l_11p = lg_e(:,1);
            ind_g_11d = [lg_e(end/2+1:end,2)-ne_g_h;lg_e(1:end/2,2)+
                ne_g_v];
```

```
        ind_l_11d = [lg_e(end/2+1:end,1)-ne/2;lg_e(1:end/2,1)+ne/2];
        ind_g_11 = ind_g_11p;
        ind_l_11 = ind_l_11p;
        ind_g_1d1d = ind_g_11d;
        ind_l_1d1d = ind_l_11d;
        ind_g_22 = lg_v(:,2);
        ind_l_22 = lg_v(:,1);
    end
    [tmpj, tmpi] = meshgrid(ind_g_00,ind_g_00); tmpi = tmpi(:); tmpj
        = tmpj(:);
    ind = (count00:count00+length(temp_v00)-1); count00 = max(ind)+1;
    i_M00(ind,1) = tmpi; j_M00(ind,1) = tmpj; v_M00(ind,1) = temp_v00
        ; clear tmpi tmpj ind

    [tmpj1, tmpi1] = meshgrid(ind_g_11(1:end/2),ind_g_11(1:end/2));
        tmpi1 = tmpi1(:); tmpj1 = tmpj1(:);
    [tmpj, tmpi] = meshgrid(ind_g_11(end/2+1:end),ind_g_11(end/2+1
        :end)); tmpi = tmpi(:); tmpj = tmpj(:);
    tmpi = [tmpi1;tmpi]; tmpj = [tmpj1;tmpj]; clear tmpi1 tmpj1
    ind = (count11:count11+length(temp_v11)-1); count11 = max(ind)+1;
    i_M11(ind,1) = tmpi; j_M11(ind,1) = tmpj; v_M11(ind,1) = temp_v11
        ; clear tmpi tmpj ind

    [tmpj1, tmpi1] = meshgrid(ind_g_1d1d(1:end/2),ind_g_1d1d(1:end/2)
        ); tmpi1 = tmpi1(:); tmpj1 = tmpj1(:);
    [tmpj, tmpi] = meshgrid(ind_g_1d1d(end/2+1:end),ind_g_1d1d(end
        /2+1:end)); tmpi = tmpi(:); tmpj = tmpj(:);
    tmpi = [tmpi1;tmpi]; tmpj = [tmpj1;tmpj]; clear tmpi1 tmpj1
    ind = (count1d1d:count1d1d+length(temp_v1d1d)-1); count1d1d = max
        (ind)+1;
    i_M1d1d(ind,1) = tmpi; j_M1d1d(ind,1) = tmpj; v_M1d1d(ind,1) =
        temp_v1d1d; clear tmpi tmpj ind

    [tmpj, tmpi] = meshgrid(ind_g_22,ind_g_22); tmpi = tmpi(:); tmpj
        = tmpj(:);
    ind = (count22:count22+length(temp_v22)-1); count22 = max(ind)+1;
    i_M22(ind,1) = tmpi; j_M22(ind,1) = tmpj; v_M22(ind,1) = temp_v22
        ; clear tmpi tmpj ind

    [tmpj, tmpi] = meshgrid(ind_g_22,ind_g_00); tmpi = tmpi(:); tmpj
        = tmpj(:);
    ind = (count02h:count02h+length(temp_v02h)-1); count02h = max(ind
        )+1;
    i_M02h(ind,1) = tmpi; j_M02h(ind,1) = tmpj; v_M02h(ind,1) =
        temp_v02h; clear tmpi tmpj ind

    [tmpj1, tmpi1] = meshgrid(ind_g_11d(1:end/2),ind_g_11p(1:end/2));
         tmpi1 = tmpi1(:); tmpj1 = tmpj1(:);
    [tmpj, tmpi] = meshgrid(ind_g_11d(end/2+1:end),ind_g_11p(end/2+1
        :end)); tmpi = tmpi(:); tmpj = tmpj(:);
    tmpi = [tmpi1;tmpi]; tmpj = [tmpj1;tmpj]; clear tmpi1 tmpj1
    ind = (count11h:count11h+length(temp_v11h)-1); count11h = max(ind
        )+1;
    i_M11h(ind,1) = tmpi; j_M11h(ind,1) = tmpj; v_M11h(ind,1) =
        temp_v11h; clear tmpi tmpj ind

%                 M00(ind_g_00,ind_g_00) = M00(ind_g_00,ind_g_00) + M00_1(
```

```
    ind_l_00,ind_l_00);
%                 M11(ind_g_11,ind_g_11) = M11(ind_g_11,ind_g_11) + M11_l(
    ind_l_11,ind_l_11);
%                 M22(ind_g_22,ind_g_22) = M22(ind_g_22,ind_g_22) + M22_l(
    ind_l_22,ind_l_22);
%                 M02_H(ind_g_00,ind_g_22) = M02_H(ind_g_00,ind_g_22) + M02_H_l(
    ind_l_00,ind_l_22);
%                 M11_H(ind_g_11p,ind_g_11d) = M11_H(ind_g_11p,ind_g_11d) +
    M11_H_l(ind_l_11p,ind_l_11d);

            end
        end
        M00 = sparse(i_M00,j_M00,v_M00);
        M11 = sparse(i_M11,j_M11,v_M11);
        M1d1d = sparse(i_M1d1d,j_M1d1d,v_M1d1d);
        M22 = sparse(i_M22,j_M22,v_M22);
        M02_H = sparse(i_M02h,j_M02h,v_M02h);
        M11_H = sparse(i_M11h,j_M11h,v_M11h);
    else
        M00 =  M00_l;
        M11 = M11_l;
        M1d1d = M1d1d_l;
        M22 = M22_l;
        M02_H = M02_H_l;
        M11_H = M11_H_l;
    end
%     M00(abs(M00)<10^-8) = 0;
%     M11(abs(M11)<10^-8) = 0;
%     M22(abs(M22)<10^-8) = 0;
%     M02_H(abs(M02_H)<10^-8) = 0;
%     M11_H(abs(M11_H)<10^-8) = 0;

%     save([location,filename],'M00','M11','M11_H','M1d1d','M22','M02_H','P','N
    ','M','nDeriv','domain');
% end
```

B.4 HermitePoly.m

```
function [h, hp, e] = HermitePoly(xf,P,nDeriv,N,plotje,k)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% % Script to calculate the Hermite Polynomials for N elements on the interval
    [-1 1]
% interpolated at location xf using nDeriv derivatives (0 or 1) and (P+1)
% polynomials per derivative.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% % Testscript
% clearvars; close all; clc;
% P = 1;
% nDeriv = 1;
% N = 2;
% xf = linspace(0,1,1000);
% plotje = 1;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

if nargin<6
    k=1;
if nargin<5
    plotje=0;
    if nargin<4
        N=1;
        if nargin<3
            nDeriv=1;
            if nargin<2
                P=1;
            end
        end
    end
end
end

if size(xf,1)>size(xf,2)
    xf=xf';
end
ns = GLLnodes(P); % [-1 Gnodes(N-1) 1]; % linspace(-1,1,N+1); %
if size(ns,1)<size(ns,2)
    ns=ns';
end

if nDeriv>1
    method=1;
else
    method=2;
end

nx = size(xf,2);
npol = (nDeriv+1)*(P+1);

h = zeros(npol,nx);
hp = zeros(npol,nx);
```

```
% e = zeros(npol-(nDeriv+1),nx);
e = zeros(P,nx);
                                                                    56
if method==1
    A=zeros(npol);
    A(1:P+1,npol) = 1;
    for i=2:npol
        a1 = ns.^(i-1);                                             61
        if nDeriv>0
            a2 = (i-1)*ns.^(i-2);
            if nDeriv>1
                if i==2
                    a3 = zeros(size(ns));                           66
                else
                    a3 = (i-2)*(i-1)*ns.^(i-3);
                end
            else
                a3=[];                                              71
            end
        else
            a2 = [];
            a3 = [];
        end                                                         76
        A(:,npol-i+1) = [a1;a2;a3];
    end
    B = diag(ones(npol,1));
    C = A\B;
    D=zeros(size(C));                                               81

    for i=1:npol
        h = h+C(npol-i+1,:)'*xf.^(i-1);
        if i>1
            hp = hp+(i-1).*C(npol-i+1,:)'*xf.^(i-2);                86
        end
    end

elseif method==2
    [Lf,~,Lpf] = MimeticpolyVal((xf-xf(1))*N-1,P,1); % MimeticpolyVal(round((xf-   91
        xf(1))*N-1,12),P,k); %
    [~,~,Lp] = MimeticpolyVal(ns,P,k);
    ns1 = GLLnodes(P);
    ns1 = (ns1-ns1(1))/N+xf(1);
    Lpf = Lpf*N;
    Lp = Lp*N;                                                      96

    if nDeriv>0
        for i=1:P+1
            h(i,:) = (1-2*Lp(i,i).*(xf-ns1(i))).*Lf(i,:).^2;
            h(i+P+1,:) = (xf-ns1(i)).*Lf(i,:).^2;                   101
            hp(i,:) = (1-2*Lp(i,i).*(xf-ns1(i))).*(2.*Lf(i,:).*Lpf(i,:)) + Lf(i
                ,:).^2.*(-2*Lp(i,i));
            hp(i+P+1,:) = (xf-ns1(i)).*(2.*Lf(i,:).*Lpf(i,:)) + Lf(i,:).^2;
        end
    else
        h=Lf;                                                       106
        hp=Lpf;
    end
```

```
    for i=1:P
        e(i,:)=-sum(hp(1:i,:),1);
        if nDeriv>0
            e(i+P,:)=h(P+1+i,:)-h(P+2+i,:);
        end
    end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if plotje
    kleur ='brgmckybrgmckybrgmckybrgmckybrgmckybrgmckybrgmckybrgmckybrgmcky';
    scrsz = get( groot, 'Screensize' );
    figure('Position',[scrsz(3)/10 scrsz(4)/10 scrsz(3)*0.8 scrsz(4)*0.75])
    subplot(2,2,1)
    hold on
    for i=1:npol
        if i>P+1
            str{i} = ['h^1_',num2str(i-1-(P+1))];
        else
            str{i} = ['h^0_',num2str(i-1)];
        end
        plot(xf,h(i,:),kleur(i));
    end
    xlim([xf(1) xf(end)]);
    hold off
    grid
    xlabel('\xi')
    ylabel('h_i(\xi)')
    legend(str,'Location','EastOutside'); clear str;
    subplot(2,2,2)
    hold on
    for i=1:npol
        if i>P+1
            str{i} = ['hp^1_',num2str(i-1-(P+1))];
        else
            str{i} = ['hp^0_',num2str(i-1)];
        end
        plot(xf,hp(i,:),kleur(i));
    end
    xlim([xf(1) xf(end)]);
    hold off
    grid
    xlabel('\xi')
    ylabel('hp_i(\xi)')
    legend(str,'Location','EastOutside'); clear str;
    subplot(2,2,3)
    hold on
    for i=1:size(e,1)
        plot(xf,e(i,:),kleur(i));
    end
    xlim([xf(1) xf(end)]);
    hold off
    grid
    xlabel('\xi')
    ylabel('e_i(\xi)')
    if nDeriv>0
    subplot(2,2,4)
```

```
%     plot(xf,[sum(hp(end/2+1:end,:),1);sum(h(P+2:end,:),1)]);
    plot(xf,[sum(hp(1:end/2,:),1);sum(hp(end/2+1:end,:),1);sum(h(1:P+1,:),1);sum(
        h(P+2:end,:),1)]);
%     plot(xf,[sum(hp(1:end/2,:),1);sum(h(1:P+1,:),1)]);
    xlim([xf(1) xf(end)]);
    grid
    xlabel('\xi')
    ylabel('sum h_i(\xi)')
    end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%
if 0
    figure;
    for i=1:7
        for j=1:7
            subplot(7,7,j+(i-1)*7);
            surf(x,y,tmp(i,:)'*tmp(j,:),'EdgeColor','none');
            set(gca,'xticklabel',[])
            set(gca,'yticklabel',[])
            set(gca,'zticklabel',[])
        end
    end
end
```